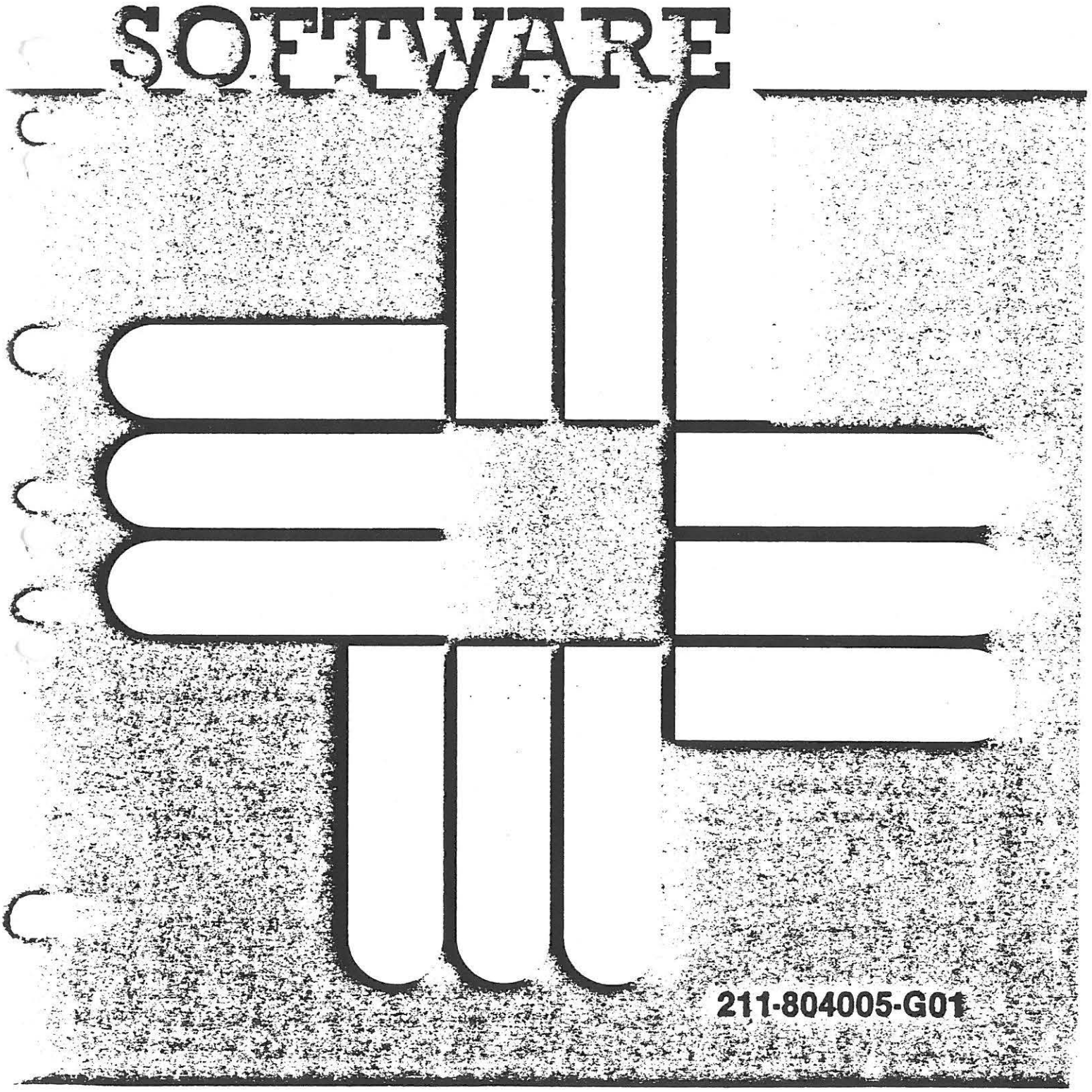
MODCOMP
MODULAR COMPUTER SYSTEMS, INC.

MAX IV/MAX 32 Programmer's Reference Manual

Source Editor (SED)

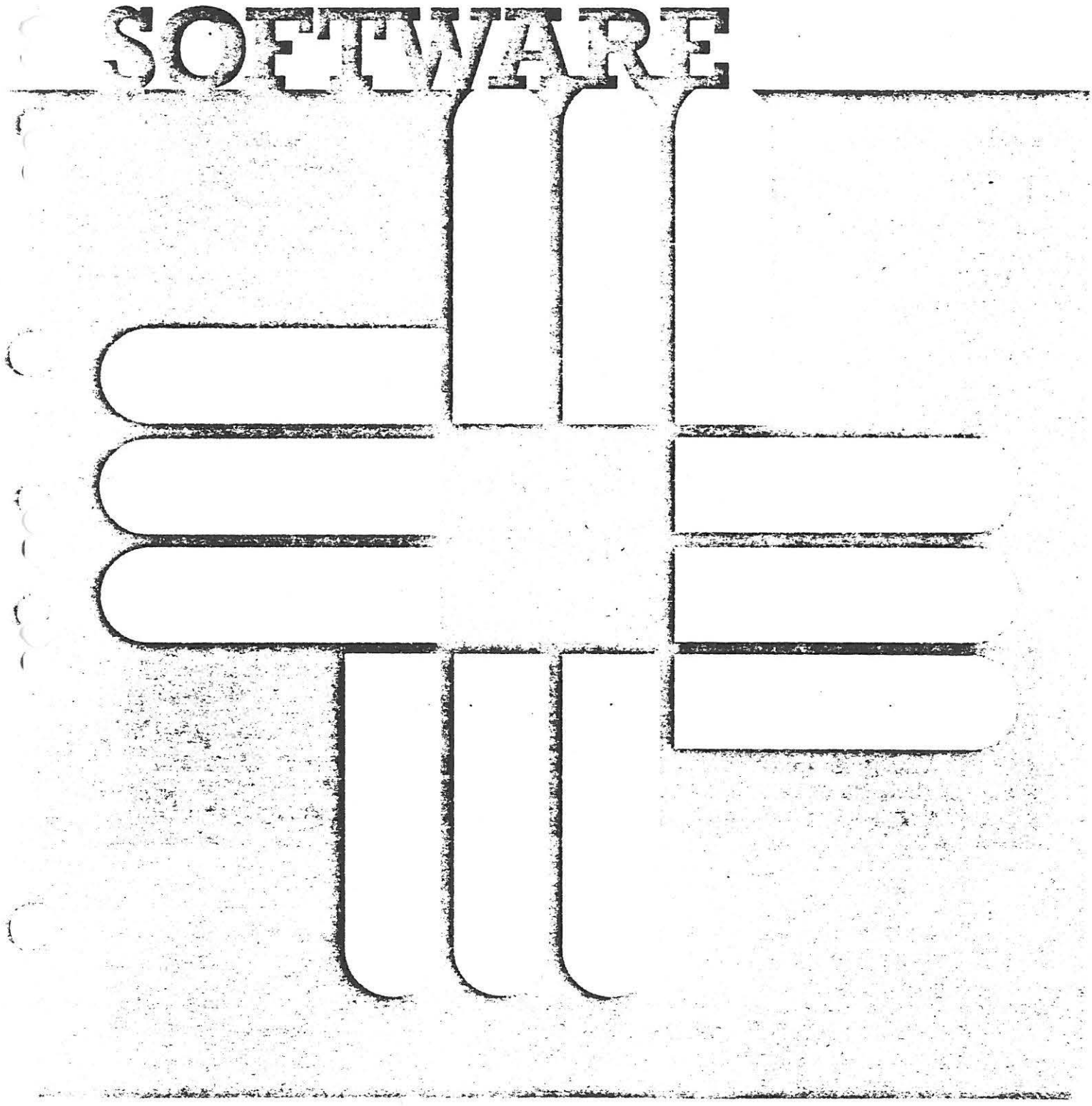


211-804005-G01

MODCOMP
MODULAR COMPUTER SYSTEMS, INC.

MAX IV/MAX 32 Programmer's Reference Manual

Source Editor (SED)



MANUAL HISTORY

Manual Order Number: 211-804005-G01

Title: MAX IV/MAX 32 PROGRAMMER'S REFERENCE MANUAL, SOURCE EDITOR

Model Number: MAX IV - 8240I MAX 32 - 8250A

Software Part Number: 611599-130G.0

Revision Level	Date Issued	Description
--	07/76	Initial Issue (A.0).
--	04/77	Revision (B.0).
--	08/77	Revision (C.0).
--	11/77	Reissue (D.0).
--	06/79	Reissue (E.0).
--	07/80	Reissue (E.2).
--	09/81	Reissue (F.0).
F01	06/84	Reissue (F.1).
G00	06/85	Revision (G.0). Added descriptions for the following new directives: EXSEQUENCE, PACK, PC, NOPACK, SCALE. The CATALOG directive accepts an asterisk as a parameter to catalog the program named in the most recent POS directive. The most recent text string specified in the SEARCH directive is retained for subsequent searches.
G01	10/85	Reissue (G.0). Manual format was reorganized to MODCOMP standards. SED is now supported under MAX 32, but there are no technical changes to the program software.

Contents subject to change without notice.

Copyright © 1985, by Modular Computer Systems, Inc.
All Rights Reserved.
Printed in the United States of America.

PROPRIETARY INFORMATION

THIS MANUAL CONTAINS CONFIDENTIAL PROPRIETARY INFORMATION PROTECTED BY SOFTWARE PROPERTY RIGHTS AND IS MADE AVAILABLE UPON THE CONDITION THAT THE SOFTWARE CONTAINED HEREIN WILL BE HELD IN ABSOLUTE CONFIDENCE AND MAY NOT BE DISCLOSED IN WHOLE OR IN PART TO OTHERS WITHOUT THE PRIOR WRITTEN PERMISSION OF MODULAR COMPUTER SYSTEMS, INC.

FOR GOVERNMENT USE THE FOLLOWING SHALL APPLY

RESTRICTED RIGHTS LEGEND

USE, DUPLICATION OR DISCLOSURE BY THE GOVERNMENT IS SUBJECT TO RESTRICTIONS AS SET FORTH IN PARAGRAPH (b) (3) (B) OF THE RIGHTS IN TECHNICAL DATA AND COMPUTER SOFTWARE CLAUSE IN DAR7-104.9(a) (OR SUCH EQUIVALENT FFR, NASFR, AND/OR DOEPR CLAUSE AS MAY BE APPLICABLE).

MODULAR COMPUTER SYSTEMS, INC.
1650 WEST McNAB ROAD
FORT LAUDERDALE, FL 33309

PREFACE

Audience
This manual is directed to programmers who want to create and maintain source programs using the MAX IV/MAX 32 Source Editor (SED). Some familiarity with either the MAX IV or MAX 32 Operating System is assumed.

Subject
This manual describes the use of SED to create, edit, store, copy, and list source programs. Chapter 1 contains general information about SED applications, a summay of commands, options, and logical files, and the procedures to access and exit SED. Chapter 2 is a tutorial for new SED users. It gives background information about the general operation of SED and reviews some of the basic command used to edit, format, list, copy, and catalog source programs. Chapter 3 is a reference containing detailed descriptions of all SED commands, including usage examples and sample output formats.

Product Requirements
SED is a component of both the MAX IV and MAX 32 Operating Systems.

Product Support
SED output can be used as input to any of MODCOMP's compilers or assemblers.

Related Publications
Refer to the following manuals for more information. When ordering manuals, use the manual order number listed below. The latest revision (REV) will be shipped.

Manual Order Number	Manual Title
213-804001-REV	MAX IV GENERAL OPERATING SYSTEM, System Guide Manual (SGM)
211-804011-REV	MAX IV AND MAX 32 TASK/OVERLAY CATALOGER, Programmer's Reference Manual (PRM)
211-804002-REV	MAX IV/MAX 32 NONRESIDENT JOB CONTROL AND BATCH FACILITIES, Programmer's Reference Manual (PRM)

MODCOMP Product Training
MODCOMP's Education Services provides training courses on many hardware and software products at our Training Center in Ft. Lauderdale, Florida. On-site courses at customer locations can also be arranged. For more information about the courses offered by Education Services, contact the MODCOMP Registrar.

Special Symbols and Notations
Special symbols and format conventions used to define the syntax for SED commands are described at the beginning of Chapter 3.

TABLE OF CONTENTS

Page

CHAPTER 1 INTRODUCTION TO SED

1.1 FUNCTIONAL DESCRIPTION 1-1
1.2 MODES OF OPERATION 1-1
1.3 SUMMARY OF SED COMMANDS 1-2
1.4 ACCESSING SED 1-8

1.4.1 SED Options 1-9

1.5 EXITING SED 1-10
1.6 LOGICAL FILES USED BY SED 1-10
1.7 ERROR RECOVERY 1-11

CHAPTER 2 A SED TUTORIAL

2.1 ORGANIZATION OF THIS CHAPTER 2-1
2.2 ENTERING SED COMMANDS 2-2
2.3 CREATING NEW SOURCE PROGRAMS

2.3.1 Building a Directory 2-3
2.3.2 Entering Data for New Programs 2-3
2.3.3 Cataloging Programs in a USL File 2-4
2.3.4 Using Tabs 2-4

2.4 GENERAL INFORMATION ABOUT EDITING UNDER SED 2-6

2.4.1 Editing in Random and Sequential Modes 2-6
2.4.2 Methods of Referencing Source Lines in SED Commands 2-10

2.4.2.1 Line Numbers 2-10
2.4.2.2 Offset from Current Line 2-11
2.4.2.3 Character String References 2-11

2.4.3 The Line Pointer 2-13

2.4.3.1 Automatically Displaying the Current Line 2-14

2.4.4 Sequential Referencing 2-15

2.5 EDITING SOURCE PROGRAMS 2-15

2.5.1 Locating Source Programs on the SI File 2-16
2.5.2 Adding Records 2-16
2.5.3 Deleting Records 2-17

Page

2.5.4 Replacing Records 2-17

2.5.4.1 Redefining the ADD/REPLACE Terminator Character 2-18

2.5.5 Duplicating and Moving Records 2-19
2.5.6 Modifying Characters within a Record 2-19

2.5.6.1 The OVERLAY Command 2-20
2.5.6.2 The CHANGE Command 2-20
2.5.6.3 The MODIFY Command 2-21
2.5.6.4 The DST, MST, and IST Commands 2-22

2.5.7 Recataloging Programs in a USL File 2-24

2.6 PRINTING, DISPLAYING, AND LISTING SOURCE PROGRAMS 2-24

2.6.1 The PRINT Command 2-24
2.6.2 The DISPLAY Command 2-25
2.6.3 The LIST Command 2-25
2.6.4 Modifying Output Listings 2-26

2.7 REMOVING AND RENAMING PROGRAMS

2.7.1 Removing Programs from a USL Directory 2-27
2.7.2 Renaming Programs 2-27

2.8 FILE MARKS 2-28

2.8.1 File Mark Output Control 2-30

2.9 COPYING SOURCE PROGRAMS 2-30

2.9.1 Copy to the SO File 2-30
2.9.2 Transfer to the SO File 2-31
2.9.3 Copying a Directory to Disc or Tape 2-32

2.10 USING MACROS 2-33
2.11 INSERT FILES 2-34
2.12 LABELS AND SEQUENCE NUMBERS 2-35

2.12.1 Protecting Label and Sequence Number Fields 2-36
2.12.2 Checking Sequence Numbers 2-36
2.12.3 Resetting Sequence Numbers 2-37

2.13 SOURCE PROGRAM VERIFICATION 2-37
2.14 DIRECTORY MAINTENANCE

2.14.1 Listing the Contents of a Directory 2-38
2.14.2 Squeezing a Directory 2-40
2.14.3 Attaching and Detaching Space from Other Partitions 2-40
2.14.4 Saving and Restoring USLs 2-41

Page

CHAPTER 3 SED COMMANDS

3.1 SPECIAL SYMBOLS AND NOTATIONS USED IN THIS CHAPTER 3-1
3.2 SED/JOB CONTROL COMMANDS 3-3
3.3 SED COMMANDS 3-4

ADD 3-5
ATTACH 3-7
BACKWARD 3-8
CAE 3-9
CATALOG 3-10
CHANGE 3-11
CHECK 3-13
CLEARTAB 3-13
COMPRESS 3-14
CONTINUE 3-14
COPY 3-15
DEFINE 3-17
DELETE 3-18
DETACH 3-19
DISPLAY 3-20
DST 3-21
DUPLICATE 3-22
ENDMACRO 3-23
EXIT 3-23
EXSEQUENCE 3-24
FILE 3-25
FORWARD 3-25
GOTO 3-26
HITHER 3-28
HOME 3-29
INITIALIZE 3-30
INSERT 3-31
IST 3-32
LABEL 3-33
LAE 3-34
LALL 3-35
LDE 3-36
LDIRECTORY 3-38
LDS 3-40
LEOF 3-41
LIMIT 3-42
LINENUMBER 3-43
LIST 3-44
LNAME, LNS, LNT 3-45
LST 3-48
MACRO 3-50
MARK 3-52
MODIFY 3-53
MOVE 3-55
MST 3-56
NGE, IGE 3-57

Page

NOCOMPRESS 3-58
NOFILE 3-59
NOPACK 3-59
NOQUICK 3-60
NOSEQUENCE 3-61
NPU, IPU 3-62
NRE, IRE 3-63
OPTION 3-64
OVERLAY 3-66
PACK 3-68
PAUSE 3-70
PC 3-71
POSITION 3-72
PRF 3-74
PRINT 3-75
PTA 3-76
QUICK 3-76
RAE 3-77
RANDOM 3-78
RECATALOG 3-79
REDEFINE 3-80
REMOVE 3-81
RENAME 3-82
REPLACE 3-83
RESTORE 3-84
REVISION 3-85
SAVE 3-86
SCALE 3-87
SEARCH 3-88
SEQUENCE 3-89
SETTAB 3-90
SIZE 3-91
SKIPTO 3-92
SOURCE 3-94
SPLIT 3-96
SQUEEZE 3-97
TEST 3-98
TEXT 3-98
TRANSFER 3-99
TWIN 3-100
UNCHECK 3-101
UNDO 3-101
UNSEQUENCE 3-102
USE 3-102
VERIFY 3-103
XLN 3-104
XRN 3-106
YON 3-106

Page

APPENDIX A ERROR REPORTING A-1
APPENDIX B DIRECTORY ORGANIZATION AND FILE FORMATS B-1

INDEX I-1

LIST OF FIGURES

2-1 SED Operation in Random Mode 2-7
2-2 SED Operation in Sequential Mode 2-9
2-3 Format of a Directory Listing 2-39

3-1 Output Format of an LAE Command 3-34
3-2 LDE Directory Listing 3-37
3-3 LDIRECTORY Directory Listing 3-39
3-4 Output Format of an LDS Command 3-40
3-5 Listing of ARGMIS with LALL in Effect 3-46
3-6 Listing of ARGPAR with LALL in Effect 3-47
3-7 LIST Command with LNT in Effect 3-47
3-8 LAE Command with LNT in Effect 3-47
3-9 LIST Command with LALL in Effect 3-49
3-10 LIST Command with LST in Effect 3-49
3-11 LIST in Source Format 3-95
3-12 LIST in Text Format 3-95

CHAPTER 1
INTRODUCTION TO SED

1.1 FUNCTIONAL DESCRIPTION

The Source Editor (SED) is a processor that is used to create, edit, store, and maintain source programs under the MAX IV or MAX 32 Operating System. SED can also be used to copy programs from one medium to another (tape-to-disc and disc-to-tape), and to obtain hard-copy listings of source programs.

NOTE: In this manual, the terms "source program" and "source file" are used synonymously.

Input to SED can be from any type of device such as an interactive terminal, magnetic tape, or disc. If input is in the form of existing files, they can be sequential files on tape, cataloged files on a disc partition that contains a SED directory, or a copy of a partition on tape.

Once source files are created, they can be copied to tape in sequential form or cataloged on a disc partition under a SED directory. Groups of source programs can be maintained in a user source library (USL). If source programs are cataloged in a USL file, they are permanently maintained on the system and can be accessed at a later time by file name. The entire USL, or specific programs from the USL, can later be copied to magnetic tape if desired.

Cataloged or uncataloged output from SED can subsequently be used as input to all MODCOMP assemblers, compilers, and other processors.

1.2 MODES OF OPERATION

SED operates in one of two modes: sequential or random. When editing in sequential mode, changes that are made to existing programs must be entered in sequence (i.e., in ascending order according to source line numbers). In this mode SED processes source input record-by-record in succession, so any additions, deletions, or modifications must be made at the appropriate point. Source line numbers are not adjusted to reflect any additions or deletions until the program is recataloged or copied to another logical file or device.

In random mode modifications to an existing source program can be entered in any order. SED automatically adjusts line numbers in the program to reflect any changes due to adding or deleting source statements.

Another difference between random and sequential mode is in the use of logical files. In random mode most commands affect logical file WRK, while in sequential mode most commands operate on logical file SI and also write to logical file SO. The effect of each command on a specific logical file is explained under the individual command descriptions in Chapter 3.

More specific information about editing programs in random versus sequential mode is in Section 2.4 in Chapter 2.

1.3 SUMMARY OF SED COMMANDS

This section presents a summary of all SED commands. Commands that are prefixed by (*) are those which are referenced in Chapter 2. That chapter is a tutorial approach to some of the most commonly used SED commands. In addition, Chapter 3 contains descriptions of each SED command in alphabetical order. Commands that are common to both SED and Job Control are explained as a group at the beginning of Chapter 3. Individual descriptions of these commands are not given, as it is assumed that SED users are already familiar with these Job Control commands.

Most SED commands can be abbreviated to their first three characters, although there are some commands that can be specified by their first character only. Acceptable abbreviations are indicated in the syntax line for each command in Chapter 3.

Editing Commands

(*) ADD	Adds new source lines to a file.
(*) DELETE	Deletes existing lines from a file.
(*) REPLACE	Replaces one or more source lines with one or more additional new lines. This same function can be accomplished with a DELETE/ADD sequence.
(*) CHANGE	Replaces characters in one or more source lines with new characters.
UNDO	Reverses the effect of the most recent CHANGE command.
(*) MODIFY	Displays a source line and allows you to modify parts of it while keeping other parts intact.
(*) OVERLAY	Overwrites existing characters on a source line based on the column positions in which the new characters are specified.
(*) DST	Defines a string that can later be inserted into a source line with an IST command or removed from the current line with an MST command.
(*) IST	Inserts the currently defined character string into a specified source line.
(*) MST	Deletes the currently defined character string from the current line.
(*) DUPLICATE	Copies one or more lines in a source file and places them after the current line. The source lines are not deleted from their original location.
(*) MOVE	Moves one or more lines in a source file and places them after the current line. The lines are then deleted from their original location. This command can only be used in random mode.
SPLIT	Splits one source line into two lines.
NPU/IPU	Identifies the source line after which new text lines are to be added. The text input for the addition must be located in a file defined by a previous PRF command. IPU performs two BKFILEs and one AVFILE on the PRF file before reading the records to be inserted.

NRE/IRE — Identifies a source line or range of lines that are to be replaced. The text input for the replacement must be located in a file defined by a previous PRF command. IRE performs two BKFILEs and one AVFILE on the PRF file before reading the replacement records.

PRF — Names the file from which SED reads text input for the NPU/IPU and NRE/IRE commands.

SEARCH — Searches for a particular character string within a source file.

(*) REVISION — Clears or defines the current revision string. If defined, the revision string is automatically inserted into every line that is added, replaced, or modified in any way.

Edit Formatting and Control

PTA — Displays the current tab settings.

SCALE — Prints a scale of numbers from 1 through 79 on the terminal and also shows where tabs are currently set.

(*) CLEARTAB — Clears one or more current tab settings.

(*) SETTAB — Sets tabs.

(*) SEQUENCE — Places a four-digit sequence number and an optional four-character label in columns 73 through 80 of each output source line.

(*) EXSEQUENCE — Places a five-digit sequence number in each output source line. The NORESET parameter can be used to prevent sequence numbers from being reset when an EOF is encountered.

(*) UNSEQUENCE — Places blanks in columns 73 through 80 of each output source line.

(*) NOSEQUENCE — Stops the effect of the SEQUENCE, EXSEQUENCE, and UNSEQUENCE commands.

(*) LABEL — Defines the column positions that should be examined for subsequent character string searches.

(*) LIMIT — Protects certain columns of every source line by preventing those fields from being modified. The LIMIT option must be also set.

(*) QUICK — Inhibits the display of source lines after an OVERLAY, CHANGE, MODIFY, IST, MST, SKIPTO, GOTO, FORWARD, BACKWARD, and line number command. Same as setting the QUICK option. QUICK is the default mode when SED is first invoked.

(*) NOQUICK — Automatically displays source lines after an OVERLAY, CHANGE, MODIFY, IST, MST, SKIPTO, GOTO, FORWARD, BACKWARD, and line number command. Same as setting the NOQUICK option.

(*) DEFINE — Redefines the character used to represent a tab, the terminator character for an ADD and REPLACE, the blank space character for a MODIFY and OVERLAY, or the delete character for a MODIFY.

PACK — Removes comments from source statements in Assembly language programs.

NOPACK — Stops the effect of the PACK command.

(*) INSERT — Names the logical file from which SED reads text input for the ADD and REPLACE commands.

(*) MARK — Defines optional program and subprogram marks that will be treated by some SED commands as end-of-file (EOF) marks.

Listing Source Files

(*) DISPLAY — Displays the current, next, or previous screenful of text to the CO file. The number of lines that comprise a screenful depends on the setting of the ONE or TWO option.

(*) LIST — In random mode, lists entire source files or a specific number of lines from the WRK file to the LO file. In sequential mode LIST can only be used to list entire source files from the SI file to the LO file.

(*) PRINT — Displays one or more lines within a file to the CO file without moving the current line pointer.

PC — Same as PRINT but also shows where control characters are located in the text.

LAE — Searches the USL directory for programs whose names match a specified sequence of characters. The optional LIST parameter can be used to list the contents of the programs if any matches are found.

(*) TEXT — When issued prior to any command that lists to the LO file, subsequent listings are written without line numbers, page headings, and titles (text format).

(*) SOURCE — When issued prior to any command that lists to the LO file, subsequent listings are written with line numbers, page headings, and titles (source format). SOURCE is the default mode when SED is first invoked.

(*) LNAME/LNS/LNT — When issued prior to a command that lists source files to the LO file, the listing will contain only those source lines that have certain predefined character strings and optionally other user-defined strings in them.

(*) XLN — When issued prior to a command that lists source files to the LO file, the listing will contain only those source lines that have certain predefined character strings and optionally other user-defined strings in them.

(*) LST — When issued prior to a command that lists source files to the LO file, the listing will contain only those source lines that have user-specified character strings in them.

(*) LALL. Stops the effect of the LNAME, LNS, LNT, XLN, and LST commands. Subsequent listings of source files to the LO file will then contain all source statements. LALL is the default mode when SED is first invoked.

Data Formatting

(*) COPY Copies source files from logical file WRK (in random mode) or SI (in sequential mode) to the SO file in compressed or uncompressed format. Can be used to copy files from disc to tape or from tape to disc.

(*) TRANSFER Copies source files from logical file WRK (in random mode) or SI (in sequential mode) to logical file SO in compressed format. Not affected by the MARK command.

NGE/IGE Copies selected records from logical file SI to logical file SO. IGE performs a BKFILE before copying the records.

HOME Writes a data buffer that is partially filled with compressed records to a logical file (normally SO).

(*) VERIFY Ensures that checksum values are correct for programs on the SI file.

(*) CHECK Ensures that sequence numbers in columns 77 through 80 of source programs on the SI file are in ascending order.

(*) UNCHECK Stops the effect of the CHECK command.

(*) COMPRESS Writes all subsequent source records to the SO file in a compact format.

(*) NOCOMPRESS Stops the effect of the COMPRESS command.

(*) FILE Writes an end-of-file mark to the SO file when one is encountered on the SI file. FILE is the default mode when SED is first invoked.

(*) NOFILE Stops the effect of the FILE command.

Positioning Within Source Files

(*) GOTO Moves the line pointer to a specific source line. In sequential mode the lines that are bypassed are copied to the SO file.

(*) SKIPTO Moves the line pointer to a specific source line. In sequential mode the lines that are bypassed are not copied to the SO file.

(*) FORWARD Moves the line pointer forward a specified number of lines. In sequential mode the lines that are bypassed are not copied to the SO file.

(*) BACKWARD Moves the line pointer backward a specified number of lines. In random mode BACKWARD operates on the WRK file. In sequential mode it operates on the SI file.

(*) LINENUMBER
Displays the line number of the current source line.

Logical File Positioning

ASSIGN	Assigns a logical file to a device or another logical file. Same as the Job Control $ASSIGN command.
(*) POSITION	Sets logical file SI to the named source file so that it can be edited, copied, listed, or examined.
AVFILE	Advances a logical file a specified number of file marks. Same as the Job Control $AVFILE command.
BKFILE	Backspaces a logical file a specified number of file marks. Same as the Job Control $BKFILE command.
AVRECORD	Advances a logical file a specified number of records. Same as the Job Control $AVRECORD command.
BKRECORD	Backspaces a logical file a specified number of records. Same as the Job Control $BKRECORD command.
REWIND	Rewinds a logical file to its beginning position. Same as the Job Control $REWIND command.
(*) LEOF	Advances logical file SI a specified number of double file marks.
WEOF	Writes an end-of-file mark on a specified logical file. Same as the Job Control $WEOF command.

Directory Handling

(*) INITIALIZE	Builds a new directory for a USL file and erases all previous data.
(*) CATALOG	Places a source file in a USL and creates a directory entry for it.
(*) RECATALOG	Replaces a cataloged source file with a newer version of the same name.
(*) SQUEEZE	Compresses a USL when its directory sectors are scattered throughout the USL file.
(*) RENAME	Changes the name of a cataloged source file. The original name can no longer be used to reference the file.
(*) TWIN	Assigns an alias name to a cataloged source file. Either name can then be used to reference the file.
(*) REMOVE	Deletes the entry for a cataloged source file from the USL directory. The file can no longer be accessed by name.

(*) RAE	Same as REMOVE but more than one entry at a time can be deleted from the directory because wildcard characters can be used when specifying the source program name.
CAE	Searches the USL directory for program names that match a specified sequence of characters and copies those programs to the SO file. Wildcard characters can be used in specifying the program name.
(*) LDIRECTORY	Lists the names of all source programs on the USL file and also gives statistical information about the use of directory sectors for programs cataloged in the USL.
LDE	Lists the names of all source programs on the USL file. Also provides optional parameters to test the programs for data integrity and if the USL file is a tape, to prevent it from being rewound.
LDS	Returns statistical information about the use of directory sectors for source programs cataloged in a USL. Also provides optional parameters to test the programs for data integrity and if the USL file is a tape, to prevent it from being rewound.
(*) ATTACH	Attaches the space from another disc partition so that it is available when source programs are cataloged in a USL.
(*) DETACH	Detaches the space of a previously attached file.
REDEFINE	Renames an attached file.
(*) SAVE	Copies a USL to the SO file in a special saved format, including the directory, all cataloged source programs, and any attached files. The SO file can be another device (i.e., to copy a USL to tape in saved format).
(*) RESTORE	Reinstates a previously saved USL (the reverse of SAVE).
SIZE	Updates the USL directory regarding information about the size of the USL file.
USE	Places an identification string into the directory of the USL file.

Miscellaneous Commands

OPTION	Sets or resets various SED options.
EXIT	Exits SED and returns to Job Control.
RANDOM	Used to enter or exit random mode or recover the most recent program on the WRK file.
XRN	Exits random mode and returns SED to sequential mode. Same as using the RANDOM command with the EXIT parameter.

$NOP	Dummy command that is ignored by SED. Usually used to insert comments into a job stream.
(*) MACRO	Defines the beginning of a macro prototype, lists or removes a specific macro, or lists the definitions of all current macros.
(*) ENDMACRO	Ends a macro definition.
(*) CONTINUE	Placed within a macro definition to cause the macro to execute continuously until the beginning-of-program (BOP) or end-of-program (EOP) is reached.
PAUSE/ACTION	Writes a message to the CO file for the operator's attention. SED then enters a hold state and execution must be resumed through the Operator Communication (OC) task. ACTION is the same as the Job Control $ACTION command.
NOTE	Same as PAUSE and ACTION except that NOTE does not cause SED to enter a hold state. After the message is displayed on the CO file, SED continues processing and reads the next command. Same as the Job Control $NOTE command.
(*) TEST	Tests the data integrity of source programs on the SI file that are in saved format.
HITHER	Saves the current line number of the WRK file and restores it when the YON command is used. Can only be used in random mode.
YON	Makes the current line in the WRK file the line number that was saved by a previous HITHER command. Can only be used in random mode.

1.4 ACCESSING SED

SED is invoked through the Job Control $EXECUTE command:

```
$EXECUTE SED [options]
```

Certain options can be set on the $EXECUTE command line while other options must be set with the OPTION command after invoking SED. The following two options can be set on the $EXECUTE command. Other options that are set within SED are described in Sectin 1.4.1.

Option	Meaning
LO	If the LO option is set and logical file LO is not assigned to your terminal, SED commands that you enter from CI or AI are also listed on the device to which logical file LO is assigned. In addition, the output from some commands (e.g., LIST and LAE) are also written to the device to which logical file LO is assigned. If LO is reset (NOLO), only insert files (if any are specified with the INSERT command) and SED commands (if LO is not assigned to your terminal) are written to the LO device. When you first invoke SED, the LO option is in effect by default.

AO	If AO is set, SED commands and source statement additions or replacements are read from logical file AI. If AO is not set, the default is to read SED commands and source input from logical file CI. Insert files that contain text input for source statement additions or replacements can still be used even if AO is set. In this case, only SED commands are read from logical file AI and source statement additions and replacements are read from the logical file named on the INSERT command.

1.4.1 SED Options

Other options that affect SED processing are specified on the OPTION command after invoking SED. These options are:

Option	Meaning
PROMPT/NOPROMPT	Displays or inhibits the display of a prompt character to indicate that SED is ready to accept input. The default prompt character is a period. **Default = NOPROMPT**
HOLD/NOHOLD	Controls whether or not SED enters a hold state after an error occurs. **Default = HOLD**
ONE/TWO	ONE suppresses the display of line numbers for source lines that are written to the CO file. TWO displays line numbers. For the DISPLAY command, a screenful of text is 20 lines if ONE is set and 10 lines if TWO is set. TWO is used with the NST option to affect the output of the PRINT command. This combination causes the line number and text of the line to be displayed on a single line. If ONE is set or if TWO and STD are set, PRINT displays the line number on one line and the text of the source line on the next line. **Default = TWO**
LIMIT/NOLIMIT	LIMIT must be set if the LIMIT command is used, which prevents certain columns of each source line from being modified. NOLIMIT allows any characters in every source line to be modified. The LIMIT command has no effect if the NOLIMIT option is set. **Default = NOLIMIT**
DATE/PGM	Saves the date and time or the program name as part of the information about a source program when it is cataloged in a USL. The PGM option uses the name as found on the first PGM or PROGRAM statement within the source file. Either the date and time or the program name is displayed under the IDENTIFICATION column of a directory listing. If the PGM option is set and there is no PGM or PROGRAM statement in the file, this field will be left blank in a directory listing. **Default = PGM**
TAB/NOTAB	Process tabs (TAB) or treat them as normal ASCII characters (NOTAB). **Default = TAB**
SCREEN/NOSCREEN	Displays output from the LIST comand a screenful at a time (SCREEN) or generates uninterrupted output to the terminal (NOSCREEN). **Default = NOSCREEN**

MET/MCO	MET terminates processing if errors occur during macro execution. MCO continues processing if errors occur during macro execution. **Default = MET**
QUICK/NOQUICK	Displays (QUICK) or inhibits the display (NOQUICK) of output lines modified by an OVERLAY, CHANGE, MODIFY, IST, or MST command or accessed by a SKIPTO, GOTO, FORWARD, BACKWARD, or line number command. The QUICK and NOQUICK commands perform the same functions as the QUICK and NOQUICK options. **Default = QUICK**
STD/NST	NST is used in conjuntion with option TWO. If the NST and TWO options are in effect **and** the record size of the CO file is 82 bytes or longer, source lines listed to the CO file by a PRINT command appear on one line instead of two. If STD is set, PRINT output to the CO file is in standard ASCII format (i.e., the line number and text of the record appear on two lines). Refer to the PRINT command for samples of these two formats. **Default = STD**

1.5 EXITING SED

The EXIT command is used to exit SED and return to Job Control:

EXIT or EXI or E

From random mode you can also use the XRN or RANDOM EXIT commands to exit random mode but still remain in SED.

1.6 LOGICAL FILES USED BY SED

The logical files used by SED are referenced throughout the remainder of this manual. Whether you are a new or experienced SED user, you should become familiar with these files so that you understand the way SED commands affect certain logical files. You should also know what the default assignments for these logical files are at your particular installation.

Logical File	Use by SED
CI	By default SED reads commands that you enter and the text input for source records that are added and replaced (using the ADD and REPLACE commands) from the CI file.
AI	If the AO option is set, SED reads commands that you enter and the text input for source records that are added and replaced (using the ADD and REPLACE commands) from the AI file instead of from the CI file.
SI	In order to update an existing source program, SED reads the program from the SI file. To update programs that are cataloged in a USL, the SI file must be assigned to the USL file.

SO	In sequential mode SED writes all of the modifications made to a source program to the SO file. Any records that were not changed must be copied over from the SI file to the SO file (in their original order) before the program is recataloged.
WRK	In random mode SED copies the program to be updated from the SI file to the WRK file, where editing commands can be entered randomly to make changes to the program.
USL	Source programs are cataloged on the USL file. If other files have been attached (with the ATTACH command), they can also be referenced.
LO	If the LO option is set, SED commands that you enter and the output from some commands (e.g., LIST and LAE) are written to the LO file.
CO	Error messages and output generated by some commands (e.g., PRINT and DISPLAY) are written to the CO file.
INSERT	If an INSERT command is used before an ADD or REPLACE command, SED reads the input addition or replacement records from the logical file named on the INSERT command instead of from logical file CI or AI.
XQZ QXZ	These two files are used internally as SED workfiles.

1.7 ERROR RECOVERY

Error messages that are returned by SED are listed in Appendix A. Occasionally, errors may occur that SED is unable to detect. For example, an improperly constructed macro that contains a CONTINUE command may put SED into mode of continuous operation that cannot be terminated normally. If an error of this type occurs, you can abnormally terminate the operation in process by activating the Operator Communication (OC) task and entering:

```
/R ABT
```

This will discontinue the current operation. You can then continue to enter other SED commands.

This can also be used if you have initiated any type of search operation or are printing a large number of source lines from a file and you decide you do not want to finish the operation.

CHAPTER 2
A SED TUTORIAL

2.1 ORGANIZATION OF THIS CHAPTER

This chapter was written as a guide for learning how to use SED to perform some basic editing functions. Items of information are presented in the order in which they are needed to perform the following functions:

- Initialize a USL Directory
- Create and Catalog Source Programs
- Edit Source Programs and Recatalog Them
- Maintain a USL Directory

If you are a new SED user, you should read this chapter in sequence because material later on in the chapter builds upon the information presented in the beginning. If you have experience using SED, you may only need to use this chapter as a reference. This chapter is not intended to describe the operation of every SED command in detail. If you need more information about any command, refer to the individual command descriptions in Chapter 3.

The organization of information in this chapter covers the following major topic areas. The numbers enclosed in parentheses correspond to section numbers where the preceding topic is discussed.

- Entering Source Input for New Programs
  These sections describe how to enter commands (2.2); build a directory (2.3.1); enter data for new programs (2.3.2); catalog programs in a USL (2.3.3); and format input data using tabs (2.3.4).

- General Information about Editing under SED
  These sections contain background information about editing under SED: operating in random and sequential modes (2.4.1); methods of referencing source lines (2.4.2 through 2.4.2.3); the current line pointer (2.4.3, 2.4.3.1); and sequential referencing (2.4.4).

- Editing Functions
  These sections describe SED commands used to: locate a program on the SI file (2.5.1); add new records to the program (2.5.2); delete records (2.5.3); replace records (2.5.4); duplicate and move records (2.5.5); modify characters within a record (2.5.6 through 2.5.6.4); recatalog programs in a USL (2.5.7); print (2.6.1), display (2.6.2), and list source programs (2.6.3); modify output listings (2.6.4); remove programs from a directory (2.7.1); and rename programs (2.7.2)

- Miscellaneous Features
  These sections describe various optional features: alternate file marks (2.8, 2.8.1); copying programs (2.9 through 2.9.3); the use of macros (2.10); insert files (2.11); and labels and sequence numbers (2.12 through 2.12.3); and verifying source programs (2.13).

- Directory Maintenance
  These sections describe SED commands related to maintaining source programs on a directory: listing the contents of a directory (2.14.1); squeezing a directory (2.14.2); attaching and detaching space from other partitions (2.14.3); and saving and restoring USLs (2.14.4).

2.2 ENTERING SED COMMANDS

SED commands are entered one per line beginning in column one. The command name is separated from one or more command parameters by a space. Except where otherwise noted, multiple parameters are separated from each other by a space or comma.

Character strings that are used as parameters must be enclosed in delimiters. Valid string delimiters include the following sets of characters:

```
() [] <> {} '' "" :: -- .. //
```

There are special cases where some delimiters cannot be used. Restrictions on the use of string delimiters used with character string references are described in Section 2.4.2.3.

Most SED commands can be abbreviated to their first three characters, although some commands can be specified with just one character. Acceptable command abbreviations are indicated in the SYNTAX section of each command description in Chapter 3. In this chapter, all commands are spelled out in their entirety.

Any parameters that are enclosed in brackets [ ] are optional. The brackets should not be specified as part of the parameter.

Other notations that are used in illustrating the syntax of each SED command in Chapter 3 are described in Section 3.1.

2.3 CREATING NEW SOURCE PROGRAMS

2.3.1 Building a Directory

In order for you to catalog source programs under SED, a directory on disc must exist for your USL file. Usually, the system administrator or whoever is in charge of your system will create directories for all users of the system. However, if you have to build your own directory, use the INITIALIZE command to do so:

INITIALIZE [integer] [identification]

If you ever use this command, you should be absolutely certain that the USL file does not already have a directory because **all previous data on the USL file will be lost.**

The integer is an optional parameter that specifies the number of source programs (entries) you expect to catalog in the directory. SED uses this value to determine the number of directory sectors to allocate for the directory. If not specified, SED builds the directory with one sector.

The identification is also an optional parameter that specifies a one- to eight-character name. If specified, this name will recorded in the USL directory and displayed as the I.D. at the top of a directory listing.

Once a directory for your USL file exists, you can create and catalog source programs, make modifications and then recatalog them, list the contents of source programs or the names of the programs in your USL directory, copy other programs from disc or tape to the USL file, and copy programs from your directory to another device.

2.3.2 Entering Data for New Programs

The following examples illustrate the typical sequence of commands that would be used to invoke SED and enter data for a new source program. The first example uses random mode; the second uses sequential mode. As you read the information in the rest of this chapter and become more familiar with the use of SED through your own hands-on experience, all of the commands used in these examples should become more meaningful.

In these two examples, the options correspond to the SED options that were listed in Section 1.4.4. in Chapter 1. In the ASSIGN statements, TY is the device name for your terminal and SCA is used as the file name for a system scratch file. Defining the uparrow (^) as the tab character is optional if you instead choose to use the standard default tab (CONTROL-I keys). If you want to use tabs to format source input, refer to Section 2.3.4, "Using Tabs." After you have finished entering the program and are ready to catalog it, read Section 2.3.3, "Cataloging Programs in a USL File."

Example 1 - Entering Data for a New Source Program in Random Mode

```
$EXE SED
OPTION ...enter options here...
DEFINE TAB :^:
ASSIGN SI TY
RANDOM
...enter data for the program,
...using ^ to request a tab...
$$
CATALOG name (1 - 8 characters)
```

Example 2 - Entering Data for a New Source Program in Sequential Mode

```
$EXE SED
OPTIONS ...enter options here...
DEFINE TAB :^:
ASSIGN SI TY SO SCA
COPY
...enter data for the program,
...using ^ to request a tab...
$$
ASSIGN SI SCA
CATALOG name (1 - 8 characters)
```

2.3.3 Cataloging Programs in a USL File

As shown in the two previous examples, the CATALOG command is used to place a new program in a USL and create an entry for that program in the USL directory. A cataloged program can be accessed at a later time to make changes to it and can serve as input to any of MODCOMP's assemblers, compilers, or other language processors.

When you enter the RANDOM command, SED takes the program you are currently entering from the SI file and puts it on your workfile (logical file WRK). An end-of-file mark ($$) signals the end of the input data. When you enter the CATALOG command, SED takes the program that is currently on the WRK file, catalogs it in your USL, and returns to sequential mode.

SED does not use the WRK file in sequential mode. By using the COPY command, SED takes the input you are entering from the SI file and writes it on the SO file. When you catalog a program in sequential mode, SED expects to find the program to be cataloged on the SI file. This is why you must reassign the SI file after you enter the input for a new program (as shown in Example 2 above).

The name used to catalog a program can be one to eight characters in length. Any characters from the Compressed Alphanumeric (CAN) set can be used. CAN-code characters include the digits 0 - 9, letters A - Z, the dollar sign ($), period (.), colon (:), and blank space. When you catalog a program, SED records the date and time of the catalog if the DATE option is set, or the name found on the first PGM or PROGRAM statement within the source program if the PGM option is set (the default). Either the date and time or the program name is displayed when you list the contents of a directory as explained in Section 2.14.1. You can also override the date and time or program name that is cataloged in the USL directory by specifying an optional identification string with the CATALOG command. If you choose to do this, refer to the CATALOG command description in Chapter 3 for more information.

2.3.4 Using Tabs

When you are entering data for a new program, adding records to an existing program with the ADD command, replacing records with the REPLACE command, or overlaying characters with the OVERLAY command, you can use tabs to position text in certain columns. A tab is generated by pressing the CONTROL and I keys simultaneously. By default, SED sets tabs at columns 7, 21, 36, and 73, which are also the default settings for FORTRAN statements.

For example, if you enter the source line shown below, where HT represents you pressing the CONTROL-I keys:

```
HTLDM,6HTADDRHTCOMMENT
```

SED will interpret it as:

```
␣␣␣␣␣␣LDM,6␣␣␣␣␣␣␣␣␣␣ADDR␣␣␣␣␣␣␣␣␣␣␣␣COMMENT
```

where ␣ represents a blank space.

If you do not want to use the default tab settings, you can clear all of the current tab settings with:

```
CLEARTAB ALL
```

To clear specific tab settings, use:

```
CLEARTAB integer [integer] [integer] ...
```

To set your own tabs, use:

```
SETTAB integer [integer] [integer] ...
```

You can also redefine the tab character to another character if you do not want to use the CONTROL-I keys to request a tab. This is done with the DEFINE command:

```
DEFINE TAB (character)
           #integer
```

If you define the tab as a character, it must be a single character that is enclosed within any of the valid string delimiters that were listed in Section 2.2. If you define the tab as a hexadecimal integer, the tab character will be the ASCII representation for that value. For example:

```
DEFINE TAB :^:
```

sets the tab character to an uparrow (^), while:

```
DEFINE TAB #27
```

sets the tab character to the ASCII representation of hexadecimal 27, which is an apostrophe ('). The value of the default tab (CONTROL-I) is hexadecimal 09.

Tabs can only be used when you are entering source records for a new program, adding and replacing records with the ADD and REPLACE commands, and when modifying records with the OVERLAY command, but not when you are changing text with a CHANGE or MODIFY command.

2.4 GENERAL INFORMATION ABOUT EDITING UNDER SED

The following sections (2.4.1 through 2.4.4) contain background information about editing source programs using SED. Many of the terms and concepts discussed here are used throughout subsequent sections that explain the individual SED commands used to perform specific editing functions.

2.4.1 Editing in Random and Sequential Modes

It is easier to edit a source program in random mode, especially for new SED users, because editing commands can be entered in any order. In addition, SED automatically renumbers the source program as you edit to accommodate any new lines that are added or any lines that are deleted.

When you first enter random mode after positioning to the appropriate program on the SI file, SED copies the original program from SI to your work file (logical file WRK). All changes that you make to the source program are made directly on the WRK file. After you have made all of the desired changes, you can then copy the newly modified program to the SO file or recatalog it in your USL directly from the WRK file.

The following example shows a typical sequence of commands that could be used to edit a program named PGM2 in random mode and then recatalog it in your USL. It is assumed that PGM2 is already cataloged as a directory entry in your USL file.

```
$EXECUTE SED
OPTION ..enter options here...
ASSIGN SI USL
POSITION PGM2
RANDOM
...enter SED editing commands here...
...commands can be entered in any order...
...when you have finished making all changes, enter...
RECATALOG PGM2
```

The diagram in Figure 2-1 illustrates what is happening as the above commands are processed by SED. The commands being entered from the CI file (or the AI file is the AO option is set) are listed in the left-hand column. The boxes show the effect of each command on the appropriate logical file. As you enter commands, they are also listed on the device that is assigned to logical file LO if it is not your terminal. Output from various commands, such as LIST, LAE, and LDIRECTORY, are also written to the LO file, while the output from other commands, such as PRINT and DISPLAY, are written to the CO file. Normally the LO and CO files are assigned by default to your terminal, but you should explicitly find out what the default assignments are for all of the logical files used by SED. At some point, you will probably want to reassign the LO file to a line printer device to obtain hard-copy listings of your programs.

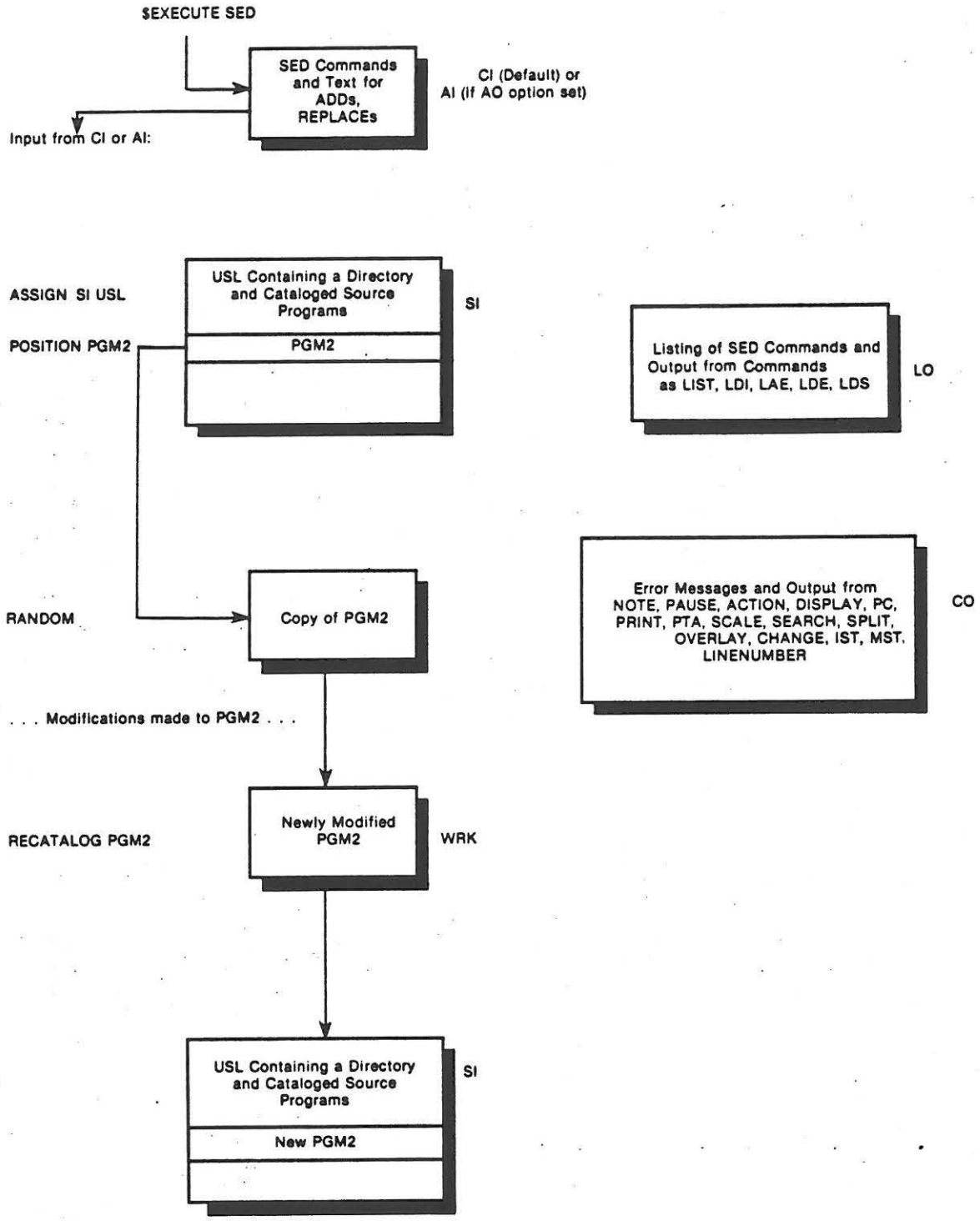

$EXECUTE SED
SED Commands and Text for ADDs, REPLACEs
CI (Default) or AI (if AO option set)
Input from CI or AI:
ASSIGN SI USL
USL Containing a Directory and Cataloged Source Programs
SI
POSITION PGM2
PGM2
Listing of SED Commands and Output from Commands as LIST, LDI, LAE, LDE, LDS
LO
Error Messages and Output from NOTE, PAUSE, ACTION, DISPLAY, PC, PRINT, PTA, SCALE, SEARCH, SPLIT, OVERLAY, CHANGE, IST, MST, LINENUMBER
CO
RANDOM
Copy of PGM2
. . . Modifications made to PGM2 . . .
RECATALOG PGM2
Newly Modified PGM2
WRK
USL Containing a Directory and Cataloged Source Programs
SI
New PGM2

Figure 2-1. SED Operation in Random Mode

In sequential mode all editing commands must be entered in the order in which the original source records appear on the SI file (i.e., sequentially). Unlike random mode, the original source program is not copied from the SI file to the WRK file. Instead, as you make changes to a program, only those source lines that are affected are written to the SO file. After you have made all of the desired changes to the program, the remaining lines that have not been affected by any changes must be copied over from the SI file to the SO file, where the modified lines have been recorded. Before recataloging the program in a USL, you must reassign SI to read the program from whatever file SO was assigned to.

The following example shows a typical sequence of commands that could be used to edit a program named PGM2 in sequential mode and then recatalog it in your USL. In this example, the SO file is assigned to a system scratch file named SCA. If your system does not have a scratch file with this name, you can substitute the name of a scratch file that does exist on your system.

```
$EXECUTE SED
OPTION ...enter options here...
ASSIGN SI USL SO SCA
POSITION PGM2
...enter SED editing commands here...
...commands must be entered in sequential order...
...when you have finished making all changes, enter...
COPY
ASSIGN SI SCA
RECATALOG PGM2
```

The diagram in Figure 2-2 illustrates exactly what is happening as the above commands are processed by SED. As you make changes to source lines within PGM2, SED copies the affected lines to the SO file (which was assigned to a scratch file). For example, suppose the first change you want to make is to line 6 and you use the MODIFY command:

```
MODIFY 6
```

SED will then copy lines 1 through 5 to the SO file. After you modify line 6, suppose you want to add three source lines after line 12. You would then enter:

```
ADD 12
```

and SED would copy lines 7 through 12 to the SO file. As you enter the new lines, SED writes them to the SO file.

In sequential mode you must always use the line numbers from the original program as it exists on the SI file (i.e., as it exists before you make any changes). SED does not update the line numbers until the program is recataloged. Thus, even though you have added three new lines after line 12, line 13 on the SI file is still the original line 13, not the first new line you just added.

You would then continue to enter SED commands in the order in which the lines appear on the SI file. Assume that PGM2 is 48 lines long and the last change you made was to line 33. To copy the remaining lines of PGM2 to the SO file, you would use the COPY command without any parameters. Once PGM2 is completely on the SO file, you would then want to recatalog it to replace the previous version of it in your USL. In sequential mode the RECATALOG command reads from the SI file. Therefore the assignment for SI must be changed so that SI points to the complete program on the SO file that you have just modified.

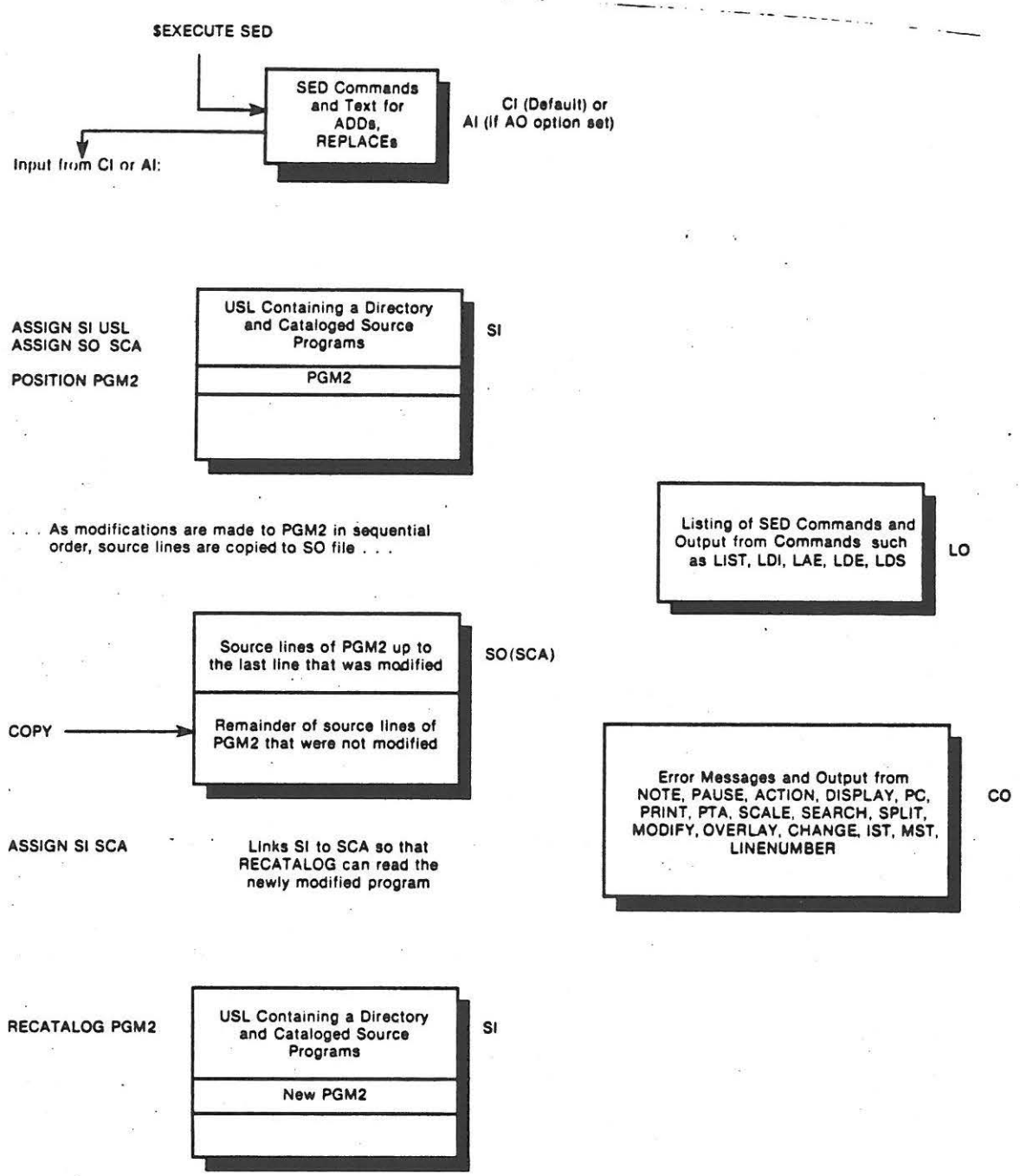

$EXECUTE SED
SED Commands and Text for ADDs, REPLACEs
CI (Default) or AI (if AO option set)
Input from CI or AI:
ASSIGN SI USL
ASSIGN SO SCA
POSITION PGM2
USL Containing a Directory and Cataloged Source Programs
PGM2
SI
. . . As modifications are made to PGM2 in sequential order, source lines are copied to SO file . . .
Listing of SED Commands and Output from Commands such as LIST, LDI, LAE, LDE, LDS
LO
Source lines of PGM2 up to the last line that was modified
Remainder of source lines of PGM2 that were not modified
SO(SCA)
COPY
Error Messages and Output from NOTE, PAUSE, ACTION, DISPLAY, PC, PRINT, PTA, SCALE, SEARCH, SPLIT, MODIFY, OVERLAY, CHANGE, IST, MST, LINENUMBER
CO
ASSIGN SI SCA
Links SI to SCA so that RECATALOG can read the newly modified program
RECATALOG PGM2
USL Containing a Directory and Cataloged Source Programs
New PGM2
SI

Figure 2-2. SED Operation in Sequential Mode

2.4.2 Methods of Referencing Source Lines in SED Commands

Most SED editing commands use a parameter that somehow identifies the line(s) to be affected by the command. For example, to add new records you would have to specify the line after which you want the new records to be inserted. In this manual that parameter is called a Source Line Reference, or SLR. Most commands use the current line as the default if you do not specify an SLR parameter. However, in some commands one or more SLR parameters are required.

SLRs can be specified in one of three forms:

- A line number
- An offset from the current line
- A character string within a record

Each type of SLR is described separately in the next three sections.

2.4.2.1 Line Numbers

In random mode line numbers change dynamically as you are editing a source program. In sequential mode line numbers remain static while you are editing (i.e., the numbers remain as they exist in the original source program on the SI file). In sequential mode line numbers are not adjusted to accommodate any additions, deletions, or replacements until the source program is copied from the SO file to another logical file or until it is recataloged in your USL. Because you have to refer to the original line numbers if you are editing a program in sequential mode, you should first obtain a hard-copy listing of the program with line numbers by using the LIST command (refer to Section 2.6.3).

In both random and sequential modes, a line number reference is an unsigned integer. For example:

```
ADD 12
```

is a line number reference to line 12. This particular command is a request to insert new lines after line 12.

While you are editing a program in SED, an imaginary line pointer is always set to some record in the source program, called the current line. The current line is generally the record that is affected by the most recent command. In random mode the current line refers to a line on the WRK file. In sequential mode the current line refers to a line on the SI file. There are some commands that have no effect on the position of the line pointer, in which case the current line is the same as it was before the command was processed. The effect of each particular command on the line pointer is noted in the individual command descriptions in Chapter 3.

A line number reference without an accompanying command moves the line pointer to that line. For example:

```
12
```

makes line 12 the current line. In random mode the line can be anywhere on the WRK file. In sequential mode the line must be located somewhere after the current line on the SI file.

There are other commands that are used explicitly to move the line pointer to another line. These commands are described later in this chapter (refer to Section 2.4.3, "The Line Pointer").

If you lose track of which line is the current line, use the LINENUMBER command.

2.4.2.2 Offset from the Current Line

A source line reference can also be in terms of the current line or some offset from the current line (either in the forward or backward direction). Just as a line number can be used to identify a source line, the following references relative to the current line can be used:

C	references the current line
C+4	references four lines forward from the current line
C-8	references eight lines before the current line

Using this type of referencing:

ADD C	could be used to add records after the current line
DEL C+8	could be used to delete the record that is eight lines after the current line

Some commands that require an SLR parameter default to the current line if you do not explicitly specify an SLR. For example:

```
ADD C
```

is the same as entering:

```
ADD
```

Although referencing source lines in relation to the current line is allowed in both random and sequential modes, it is normally used in random mode only because of the various effects different commands have on the position of the current line pointer. In addition, because you must reference source lines on the SI file in order in sequential mode, once you pass a line, you cannot reference it again. Thus, a reference with a negative offset from the current line (e.g., C-8) would cause an error in sequential mode.

2.4.2.3 Character String References

If you do not have a hard-copy listing of the source program you are editing and do not know the line numbers of the records, you can specify a character string as a source line reference. With this method, you are identifying a particular source line by a unique character string that appears on the line. A character string SLR has the form:

```
[integer1] (character string) [integer2]
```

This type of reference refers to the source line that is "integer2" lines before or after the line that contains "character string" starting in column "integer1." If "integer2" is preceded by a minus sign, it references source lines before the record that contains "character string." If "integer2" is unsigned or preceded by a plus sign, it references source lines after the record that contains "character string." The search for the specified string always starts on the current line.

If a starting column number is specified by the integer1 parameter, the delimiters used to enclose the character string cannot be slashes (/). Any other valid character string delimiters that were listed in Section 2.2. can be used.

Some examples of commands with character string references are:

ADD 4(LDI,3)-2 Allows you to insert new lines two lines before the first line encountered that containst the string "LDI,4" starting in column four.

REPLACE "ABC","ABC"+2
Deletes the range of lines from the first line encountered that contains the string "ABC" through the next two lines and allows you to insert new lines in their place.

SEARCH 7'PGM' Locates the first line that contains the string "PGM" starting in column seven.

To limit the range of columns in each source line that should be searched for a character string reference, you can predefine a range of columns with the LABEL command:

LABEL column1-column2

or

LABEL column1 column3

If the first form is used, SED only checks in "column1" through "column2" for any character string that is subsequently used as a source line reference. If the second form is used, SED only checks starting in "column1" for a length of "column3." To override the effect of a previous LABEL command, use LABEL without specifying any parameters.

To reference a source line that contains a specific character string anywhere within the line, do not give any values for "integer1" and "integer2" in the character string reference and make sure that the LABEL command is set to its default (enter LABEL without any parameters).

Note that a character string reference as an SLR parameter and the use of the LABEL command as described here is not used if you only want to search for a character string without performing some other operation based on the occurrence of that string. To search for a particular character string, use the SEARCH command. The character string reference being described in this section is used to perform some operation on a source line based on the occurrence of a character string. Any command described in Chapter 3 that has a parameter named SLR can take a character string as a source line reference.

Reviewing the different ways to reference a source line, all of the following examples are valid specifications for any SED command that has an SLR parameter:

28	A line number reference
C	The current line
C+6	A reference by positive offset from current line (the line that is six lines after the current line).

C,C+6	A range of lines based on the current line (the range of lines from the current line through the next six lines).
C-11	A reference by negative offset from current line (the line that is 11 lines before the current line).
/C/-11	A character string reference with negative offset from the line on which the string is found (the line that is 11 lines before the first line encountered that contains the character C).
5'C'+6	A character string reference with starting column number and positive offset from the line on which the string is found (the line that is six lines after the first line encountered that contains the character C in column five).
(ABC)	A character string reference where the character string can start in any column (the first line encountered that contains the string ABC).
(ABC),(ABC)+3	A range of lines based on a character string (the range of lines from the first line encountered that contains the string ABC through the next three lines).

2.4.3 The Line Pointer

While some SED commands have no effect on the position of the current line, other commands reset the line pointer after the command is processed. In sequential mode it is especially important to keep track of the position of the current line because jumping around in the program will cause errors and/or unpredictable results to the program when it is recataloged. In random mode the effect of changing the position of the line pointer is not of such concern because SED adjusts the line numbers as you are editing to accommodate any changes made to the program.

As described in Section 2.4.2.1, the line pointer can be moved to a specific source line by entering the line number as a command. For example:

```
15
```

moves the line pointer from its current position to line 15. Line 15 then becomes the current line. In sequential mode a line number reference must specify a line that is after the current line on the SI file. In random mode a line number reference can be used to move the pointer to any line within the WRK file.

There are also other commands that can be used to move the current line pointer:

```
GOTO
SKIPTO
FORWARD
BACKWARD
```

GOTO and SKIPTO have the same format:

```
GOTO SLR

SKIPTO SLR
```

SLR is any type of source line reference (line number, offset from current line, or character string) as previously described in Section 2.4.2.1 through 2.4.2.3. In random mode GOTO and SKIPTO can be used to locate a source line before or after the current line on the WRK file. In sequential mode the line you want to go to or skip to must be located after the current line on the SI file. In sequential mode the lines that are bypassed on the SI file are copied over to the SO file if GOTO is used, but not copied if SKIPTO is used.

More information about the searching patterns used by these commands is in the command descriptions in Chapter 3.

The FORWARD command moves the pointer on the WRK file (in random mode) or on the SI file (in sequential mode) forward by a specified number of lines:

FORWARD [integer]

If the number specified is more than the number of remaining lines in the program, the pointer is set to the end-of-program (EOP). If an integer is not specified, the pointer on the appropriate logical file is moved forward one line.

To move the line pointer backward by a specified number of lines, use:

BACKWARD [integer]

If the number specified is more than the number of preceding lines in the program, the pointer is set to the beginning-of-program (BOP). If an integer is not specified, the pointer is moved backward one line.

In random mode a GOTO, SKIPTO, FORWARD, and BACKWARD command repositions the line pointer to the specified line in the WRK file. In sequential mode a FORWARD, BACKWARD, or SKIPTO command causes SED to write the new current line to the SO file and then reposition the SI file without further affecting the SO file; that is, no other lines from the program are written to the SO file except the new current line. If GOTO is used in sequential mode, SED copies any source lines that are bypassed on the SI file to the SO file until it reaches the new current line. That line then becomes the current line on the SI file.

2.4.3.1 Automatically Displaying the Current Line

The NOQUICK command or option is used to automatically display the current line to the CO file after certain commands: CHANGE, MODIFY, OVERLAY, IST, MST, GOTO, SKIPTO, FORWARD, BACKWARD, and entering a line number as a command. This mode is set with either the NOQUICK or OPTION command:

NOQUICK

or

OPTION NOQUICK

To suppress the automatic display of source lines after these commands, use the QUICK command or set the QUICK option. QUICK is the default mode when you first invoke SED.

2.4.4 Sequential Referencing

In order to modify selected source lines from a program in sequential mode, references to those lines must be made in the order in which the original records appear on the SI file. This is why it is recommended to have a hard-copy listing of the program you are editing if you are working in sequential mode. For example, you cannot reference line 3 after having referenced line 5. If you do, the command in error is not processed and SED writes the following error message to the CO file:

```
! CONTROL STATEMENT ERROR
```

The following illustrates a sequence of commands that would be valid in sequential mode. The right-hand column of text describes the effect of these commands on the SO file. The line numbers in the descriptions refer to the orginal line numbers on the SI file.

ADD 1	SED copies line 1 to the SO file and also writes the new lines that are added on the SO file.
REPLACE 5,7	SED copies lines 2 through 4 to the SO file and writes the new lines that replace lines 5 through 7 on the SO file.
DELETE 9	SED copies line 8 to the SO file.
ADD 11	SED copies lines 10 and 11 to the SO file and also writes the new lines that are added on the SO file.

In the following example the DELETE 2 command would cause an error in sequential mode because line 2 precedes line 5, which was already referenced:

ADD 5	SED copies lines 1 through 5 to the SO file and also writes to the SO file any new lines that are added.
DELETE 2	SED returns a ! CONTROL STATEMENT ERROR because line 2 has already been written to the SO file.

The following DELETE command is unacceptable in both random and sequential modes because the character string reference "abc" refers to the same source line. The range of lines indicated by "abc + 7" is already included in the range of lines indicated by "abc + 10."

```
DELETE 'abc'+10,'abc'+7
```

2.5 EDITING SOURCE PROGRAMS

Once you have created source programs and cataloged them in a USL, it is likely that you will later want to access them to make corrections or modifications. The following sections (2.5.1 through 2.5.8) describe the commands used to locate programs on the SI file and edit them. The editing commands covered in these sections enable you to add, delete, replace, and move lines, modify specific characters within lines, recatalog the most recent version of a program, and list source programs while you are editing or after you have finished an editing session.

2.5.1 Locating Source Programs on the SI File

The POSITION command is used to locate source programs on the SI file. In order to locate a specific program in a USL, the SI file must be assigned to your USL file as shown in the command stream examples in Section 2.4.1. Once this assignment is made, use the POSITION command to specify the name of the program as it was originally cataloged:

```
POSITION name
```

SED searches for the specified name in the USL directory and sets the pointer to the first source line in the program. If you then enter random mode using the RANDOM command, SED copies the program to the WRK file. In sequential mode the original program remains on the SI file and selected source lines are copied from the SI file to the SO file as you edit.

When you position to a program, SED detects the end of the program by locating a file mark. By default a file mark is two dollar signs ($$). You also have the option of defining your own file marks with the MARK command, which is explained later in this chapter in Section 2.8. If you do use a file mark other than the default, you should be aware of the manner in which various SED commands, such as the POSITION, process a file mark. This information would be noted in the descriptions of the individual commands in Chapter 3.

If the source programs on the SI file are in sequential format, other parameters can be used with the POSITION command to search the SI file sequentially until a specific number of file marks are encountered. If the programs on the SI file are not cataloged in a USL directory, refer to the POSITION command description in Chapter 3 for more information on how to locate them.

2.5.2 Adding Records

To insert new lines of text into a source program, use the ADD command:

```
ADD [SLR]
```

SLR is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. If not specified, SED assumes that you want to add new lines after the current line. After entering all of the new lines, enter a point (.) or file mark ($$) on a new line starting in column one to indicate the end of the new input.

In random mode SED inserts the new lines directly into the WRK file and sets the current line pointer to the last new line that was added.

In sequential mode SED copies the original source records from the current position of the SI file to the SO file, up to the line after which the new lines are to be inserted. After the new lines are added and the add is terminated by . or $$, the current line pointer on the SI file is set to the source line that precedes the inserted records (i.e., the line that was specified in the ADD command).

To add records before line one, use:

```
ADD 0
```

By default SED expects you to enter the new lines from the CI file. This can be changed to the AI file by setting the AO option with the OPTION command. To read the input records from a logical file other than CI or AI, use the INSERT command (refer to Section 2.11 or the INSERT command description in Chapter 3 for more information).

2.5.3 Deleting Records

To delete records from a source program, use the DELETE command:

```
DELETE [SLR1 [SLR2]]
```

SLR1 and SLR2 are source line references as previously described in Sections 2.4.2.1 through 2.4.2.3. If both parameters are specified, the range of lines from SLR1 through SLR2 are deleted. If neither parameter is specified, the current line is deleted. Individual lines other than the current line can only be deleted by specifying one line at a time on separate DELETE commands. For example:

```
DEL 9,13
```

deletes lines 9 through 13, while:

```
DEL 9
DEL 13
```

deletes line 9 and 13. Keep in mind that in random mode, SED renumbers the lines on the WRK file as you add, delete, and replace lines. Therefore, after deleting line 9, the WRK file would be renumbered and the line that was originally line 13 would now be line 12. Before you use multiple DELETE commands in sequence to remove selected lines, be sure to verify that the line is really the one you want to delete.

In sequential mode SED always uses the line numbers as they exist in the original program on the SI file, so two DELETE commands in a row would remove the appropriate lines. When DELETE is used in sequential mode, SED copies the original source lines from the current position of the SI file to the SO file up to the first line to be deleted. After bypassing the line(s) to be deleted, the current line pointer on the SI file is then set to the last line that was deleted.

2.5.4 Replacing Records

To delete one or more lines in a source program and replace them with new lines, use the REPLACE command:

```
REPLACE [SLR1 [SLR2]]
```

SLR1 and SLR2 are source line references as previously described in Sections 2.4.2.1 through 2.4.2.3. If both parameters are specified, the range of lines from SLR1 through SLR2 are deleted and you are placed in insert mode to enter the replacement records. If neither line is specified, the current line is deleted and you can then enter new lines in its place.

A REPLACE is the same as deleting one or more lines and then adding new lines in the same place. Just as with an ADD, a REPLACE must be terminated by a point (.) or EOF mark ($$) on a new line starting in column one.

In random mode SED removes the specified lines directly from the WRK file and inserts the new lines in their place. The current line pointer is then set to the last new line that was added as a replacement.

In sequential mode SED copies the original source records from the current position of the SI file to the SO file up to the first source line to be replaced. After the replacement is terminated by . or $$, the current line pointer on the SI file is then set to the last line that was replaced (i.e., not the new lines that were added).

Just as with the ADD command, SED expects to read the replacement records after a REPLACE command from the CI file. This can be changed to the AI file by setting the AO option. To read the replacement records from a logical file other than CI or AI, use the INSERT command (refer to Section 2.11 or the INSERT command description in Chapter 3 for more information).

2.5.4.1 Redefining the ADD/REPLACE Terminator Character

When inserting new lines into a program using the ADD or REPLACE command, a point (.) is used to indicate that you are finished entering new input. However, this prevents you from entering a regular source line that begins with a point. To do this you can redefine the character used to terminate source record additions or replacements with the DEFINE command:

```
DEFINE TERMINATOR (character)
                  #integer
```

If an integer is specified, it must be a one- or two-digit unsigned hexadecimal number that represents the desired character's ASCII code. The # must be entered exactly as shown. If a character is specified, it must be a single character enclosed in any of the valid string delimiters that were listed in Section 2.2. After defining a new terminator, that character will be valid in all instances where a point (.) was previously valid.

NOTE: You can still use the EOF mark ($$) to terminate input after an ADD or REPLACE command even if you define a new terminator character.

The DEFINE command can also be used to define several other special characters:

- The character used to request a tab when entering new source lines or when modifying source lines with the OVERLAY command (default = CONTROL-I keys).
- The character used to insert a blank space when modifying source lines with the MODIFY and OVERLAY commands (default = &).
- The character used to delete characters when modifying source lines with the MODIFY command (default = _ (an underscore)).

Refer to the description of the DEFINE command in Chapter 3 for more information and examples of defining other characters for these functions.

To list the current definitions of the tab, terminator, blank space, and delete characters, use DEFINE without any parameters. The current definitions will be displayed on the CO file.

2.5.5 Duplicating and Moving Records

Two commands are available to move source lines from one location to another within a source program: DUPLICATE and MOVE.

DUPLICATE copies one or more lines and places them after the current line but does not remove them from their original location. DUPLICATE can be used in either random or sequential mode. With MOVE, the source lines are deleted from their original location after they are moved. MOVE can only be used in random mode.

The syntax of the DUPLICATE command is:

DUPLICATE [SLR1 [SLR2]]

SLR1 and SLR2 are source line references as previously described in Sections 2.4.2.1 through 2.4.2.3. If both parameters are specified, SED duplicates the range of lines from SLR1 through SLR2 and places them after the current line. If neither parameter is specified, SED duplicates the current line.

In random mode SED locates the specified line or range of lines on the WRK file and inserts them after the current line. The line pointer is then set to the last line that was inserted.

In sequential mode SED locates the original line or range of lines on the SI file and writes them to the current position of the SO file. The line or range of lines to be duplicated can be located before or after the current line, but the SLR parameters must refer to the line numbers as they exist on the SI file. The line pointer is not affected by DUPLICATE in sequential mode so it remains in the same position as before the command was processed.

MOVE is similar to DUPLICATE except that it deletes the lines after they are moved and can only be used in random mode:

MOVE [SLR1 [SLR2]]

After a MOVE the line pointer is set to the last line that was moved in its new location on the WRK file.

2.5.6 Modifying Characters within a Record

Rather than replacing an entire source line when only a few characters need to be changed, there are several SED commands that can be used to modify selected characters while leaving the rest of the line intact. These commands are:

OVERLAY
CHANGE
MODIFY
IST and MST (used in conjuntion with DST)

If the NOQUICK command or option is in effect, SED will also display the lines affected by these commands after they are changed.

2.5.6.1 The OVERLAY Command

OVERLAY is generally used when the number of characters to be replaced is equal to the number of characters in the replacement string and the replacement is to be made based on character position. The command is specified as follows:

OVERLAY (character string) [integer]

The character string consists of the replacement characters in their respective column positions. For example, if you only want to replace characters in the second and third columns, then the first character of specified string must be a blank space so that column one will not be overwritten. The overlay takes place only on the current line if the optional integer parameter is not specified. If an integer is specified, the overlay takes place on that number of lines, starting on the current line.

As an example, suppose that the current line of the source file you are editing contains a misspelled word:

C TIHS IS A COMMENT.

To correct the string "TIHS" to "THIS" using the OVERLAY command, you would specify the following:

OVERLAY (ßßTHIS)

where ß represents a blank space. Note that because the character "T" in the string "TIHS" starts in column three, two blank spaces are required to skip over columns one and two.

To overlay an actual blank space, use an ampersand (&) in the column position where it is to appear. For example:

OVERLAY (&&ABC)

would overlay columns one and two of the current line with blank spaces and columns three, four, and five with the string ABC.

If you want the & to appear as an actual character, you must first redefine the space character with the DEFINE command. Refer to the DEFINE command description in Chapter 3 for more information.

The tab character can also be used with the OVERLAY command. A tab has the same effect as spacing over the characters that are to remain intact with the space bar. Refer to the OVERLAY command description in Chapter 3 for an example of OVERLAY using tabs.

2.5.6.2 The CHANGE Command

When the number of characters to be replaced is not equal to the number of characters in the replacement string, the CHANGE command can be used to modify selected characters in a source line. With CHANGE, you do not have to be concerned with specifying the replacement characters in the appropriate column positions as you do with OVERLAY.

To use the CHANGE command, you must specify the character string to be replaced and the new replacement string. The change takes place starting on the current line. The syntax of the CHANGE command is:

CHANGE (old string) (new string) [integer]

If an integer is not specified, only the first occurrence of "old string" is changed to "new string" on the current line. If an integer is specified, then the first "integer" occurrences of "old string" are changed to "new string" on the current line.

To make the same change to several source lines at one time starting on the current line, use the following form:

CHANGE (old string) (new string) integer1 integer2

This will change "old string" to "new string" as many as "integer1" times per line. SED will search for the string on "integer2" lines, starting on the current line. The last line searched will become the current line, even if the specified change is not made to that line.

If a previous CHANGE command was used and you want to use the same values for both the old and new characters strings, you can omit them from the command and the previous change will be made to the current line:

CHANGE

Any values from the integer1 and integer2 parameters are not carried over, so you must explicitly specify them again, even if you are defaulting to the previous character strings. For example:

CHANGE 2,3

would use the old and new string values from the previous CHANGE command and would make the appropriate change up to two times per source line. SED would search a total of three lines for the character string, starting with the current line. The last line searched will become the current line, even if the change is not made on that line.

2.5.6.3 The MODIFY Command

MODIFY is used instead of OVERLAY or CHANGE when the changes that are to be made to a source line are more complex. MODIFY allows you to insert, delete, and overlay characters while leaving any other characters you want intact. The syntax of the MODIFY command is:

MODIFY [SLR]

SLR is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. If not specified, SED assumes that you want to modify the current line.

When a MODIFY command is used, SED displays the source line indicated by the SLR parameter (or the current line) on the CO file. You can then overlay, delete, or insert characters by entering the new information at the appropriate column positions below the displayed line according to the following rules:

Any characters that are skipped over (with the space bar) remain intact.

To overlay characters, simply enter the new characters under the corresponding characters of the displayed line.

To delete characters, enter an underscore (_) under the corresponding characters of the displayed line.

To insert characters, enter an uparrow (^) below the character where the new text is to be inserted. The new characters will be inserted before the character where the uparrow appears. After entering the inserted text, enter another uparrow, an underscore, or a press the space bar to terminate the insert.

To replace a character with a blank space, enter an ampersand (&) under the appropriate character. To insert a blank space, you must first enter an uparrow (^) and then an ampersand. If you enter ^ and then press the space bar, that will end the insert as previously described for inserting characters.

The underscore and ampersand used to represent a delete operation or blank space character are the default characters used by the MODIFY command (an ampersand is also used in the OVERLAY command to represent a blank space character). These two characters can be changed with the DEFINE command as follows:

DEFINE UNDERSCORE (character)
#integer

and

DEFINE SPACE (character)
#integer

If a character is specified, it must be a single character that is encolsed in any of the valid string delimiters that were listed in Section 2.2. If an integer is specified, it must be a one- or two-digit unsigned hexadecimal integer that represents the desired character's ASCII code. The # must be entered exactly as shown.

The DEFINE command is also used to redefine the default tab character (as explained in Section 2.3.4) and the terminator character for the ADD and REPLACE commands (as explained in Section 2.5.4.1).

2.5.6.4 The DST, MST, and IST Commands

The DST command can be used in one of two ways. The first is to define a character string that can later be inserted into a source line with the IST command. The IST command specifies the source line where the string is to be inserted. These two commands are useful if you want to insert the same character string in many lines. The syntax of the DST command is:

DST (string)

The syntax of the IST command is:

IST [SLR]

SLR is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. If not specified, SED assumes that you want to insert the currently defined string into the current line.

When you use an IST command, SED displays the specified line on the CO file. To insert the currently defined string, enter an uparrow (^) before the character where the string should be placed. To insert the string in the current line, omit the SLR parameter from the IST command.

The second use of the DST command is related to use of the MST command. MST is used to delete a character string from the current source line. For example:

```
MST (abc)
```

deletes the first occurrence of the string "abc" from the current line. If a string is not specified, SED uses the currently defined string.

Both IST and MST make use of the currently defined string. The currently defined string is either the string that was specified in a DST command or the string that was specified in an MST command, whichever was used most recently. If MST is used with a string, that string then becomes the currently defined string and overrides any string that was specified in a previous DST command. If MST is used without a string, SED uses the value of the string from the previous DST or MST command, whichever was used most recently.

The currently defined string, whether defined in a DST or MST command, remains in effect until you either redefine it (by using another DST command or an MST command with a string) or until you exit SED. This allows you to insert or delete the same string in more than one source file.

The following examples use DST, IST, and MST commands to show the effect of each on the currently defined string.

```
DST (ABC)
IST 10
```

These two commands would allow you to insert the string ABC into line 10.

```
DST (DEF)
MST "END"
```

The DST command defines DEF as the current string. However, because MST is then used with a string, the string END will be deleted from the current line and will also become the currently defined string. If you subsequently entered:

```
IST 4
```

SED would display line 4 so that you can enter an uparrow to insert the string END into that line. If you want to make DEF the currently defined string again, you would have to enter:

```
DST "ABC"
```

For more examples on the use of DST, MST, and IST, refer to the command descriptions in Chapter 3.

2.5.7 Recataloging Programs in a USL File

After you have used a combination of the editing commands discussed in the previous sections to edit a program, you naturally want to save a copy of the most recent version to replace the older version. The RECATALOG command is used to replace an exisiting cataloged program with a newer version and is specified in the following form:

RECATALOG name

or

RECATALOG *

The name must be the same one- to eight-character name that was initially used to catalog the program. If you used a POSITION command to locate the program on the SI file before you edited it, you can specify an asterisk. SED will then recatalog the program with the name specified in the most recent POSITION command. If you recatalog a program in random mode, SED returns to sequential mode after the RECATALOG command is processed.

When recataloging, SED expects to find the program on the same file as the CATALOG command does; that is, on the WRK file in random mode or on the SI file in sequential mode. In random mode this is a simple task because the program you are editing is already on the WRK file. But in sequential mode it is your responsibility to make the appropriate assignments prior to recataloging the program.

Remember that when you edit a program in sequential mode, the original program remains on the SI file. SED only copies those source records over to the SO file that are affected by any changes that you make. If you do not make changes up to the last line of the program, the remaining source lines (from the current line on the SI file to the end of the program) should be copied over to the SO file before you recatalog the program. After this is done, the SO file should then contain the complete, newly modified program with the all source line numbers in sequential order. However, you cannot recatalog directly from the SO file because in sequential mode SED expects to find the program on the SI file. Only after you reassign SI to whatever file SO was assigned to can you successfully recatalog the new version of the program. The second example in Section 2.4.1 shows a typical sequence of commands for making the appropriate SI/SO assignments, copying records to the SO file after editing a program, and then recataloging the program in sequential mode.

2.6 PRINTING, DISPLAYING, AND LISTING SOURCE PROGRAMS

There are several commands that can be used to display the source lines in a program while you are editing or after you have finished entering editing commands. These are PRINT, DISPLAY, and LIST.

2.6.1 The PRINT Command

PRINT is used to list one or more consecutive source lines on the CO file. PRINT can be used in both random and sequential modes and has the form:

PRINT [integer]

The integer specifies the number of lines to print from the WRK file (in random mode) or the SI file (in sequential mode), beginning with the current line. If a number is not specified, only the current line is printed. The current line pointer is not affected by the PRINT command.

Depending on the record size of the CO file, PRINT displays source lines in one of two formats:

```
aaaaa zzzzzzzz
<----------first 72 bytes of 80-byte record---------->

aaaaa
<------------------80-byte record------------------>
```

where aaaaa is the line number and zzzzzzzz are the last eight characters of an 80-byte source record.

Line numbers are only printed if the TWO option is set. This is the default mode when you first invoke SED. To suppress the first line of either of the above formats, set option ONE:

```
OPTION ONE
```

If the TWO and NST options are set and the record size of the CO file is at least 82 bytes, the PRINT command will display source lines on a single line (i.e., the line number and text of the record will appear on one line).

2.6.2 The DISPLAY Command

Instead of printing a specific number of lines, you can display a screenful of text at a time, beginning at the current line, with the DISPLAY command. DISPLAY can be used in either random or sequential mode and has the form:

```
DISPLAY [keyword]
```

The keyword is one of the following: C (for CURRENT), F (for FORWARD), or B (for BACKWARD), depending on whether you want to display the current, next, or previous screenful of text. If not specified, the default is C. After a DISPLAY command, the line pointer is set to the first line that is displayed.

The number of lines defined as a screenful depends on whether option ONE or TWO is set. If option ONE is in effect, a screenful is 20 lines and line numbers are not displayed. If option TWO is in effect, a screenful is 10 lines and line numbers are displayed. If line numbers are displayed, they are listed on the same line as the text of the source line if the TWO and NST options are in effect and if the record size of the CO file is at least 82 bytes. Otherwise the line number appears above the text of the line in the same format as described for the PRINT command.

2.6.3 The LIST Command

Another way to list source programs is with the LIST command. In random mode LIST can be used to list the entire WRK file or selected lines from the WRK file. In sequential mode LIST can only be used to list one or more programs from the SI file. Output from a LIST command is written to the device to which logical file LO is assigned.

The LIST command is not affected by the setting of either option ONE or TWO but is affected by the setting of the SOURCE and TEXT commands. If the SOURCE command is in effect when LIST is used, output listings will contain a header at the top of the page giving the Source Editor revision level and product name, the date and time of the listing, and the output page number. Source line numbers will also be listed.

The format of source lines generated by a LIST command when the SOURCE command is in effect is:

```
aaaaa <------------------80-byte record-------------->
```

where aaaa is the source line number. SOURCE is the default mode when SED is first invoked. To prevent the header and source line numbers from being listed, use the TEXT command. Samples of output listings in both source and text format are illustrated in Figures 3-11 and 3-12 under the SOURCE command in Chapter 3.

To list a specific number of lines from the WRK file in random mode, use the following form of LIST:

```
LIST integer
```

The integer parameter specifies the number of lines to list, starting with the current line. The line pointer is not affected by a LIST in random mode when used with an integer parameter. If an integer is not specified, SED lists the entire WRK file. In this case, the line pointer will be set to the end-of-program (EOP) on the WRK file.

To list all of the programs in your USL directory from either random or sequential mode, assign the USL file to a file with a directory and use:

```
LIST ALL
```

In sequential mode LIST is used to list one or more programs from the current position of the SI file:

```
LIST [integer]
```

In sequential mode the optional integer parameter specifies the number of programs to list, beginning at the current position of the SI file. If not specified, the default is to list one program until a file mark is encountered.

Whenever a LIST is done from sequential mode, the current line pointer is always set to the last line of the last program that was listed. Thus, in sequential mode a succession of LIST commands (entered without parameters) can be used to list programs, one after another, from the SI file. In random mode a succession of LIST commands (entered without parameters) keeps listing the same program from the WRK file.

When LIST is used in sequential mode, the following message appears after each program:

```
TOTAL RECORDS WRITTEN = xxxx/yyyy
```

xxxx is the number of logical records written and yyyy is the number of physical records written.

2.6.4 Modifying Output Listings

Several commands can be used to modify the output that is generated by any SED command that lists source files to the LO file. These commands are LST, LNAME, LNS, LNT, and XLN. LNAME, LNS, and LNT are identical and can be used interchangeably.

If the LNAME, LNS, LNT, or XLN commands are used without parameters, then subsequent listings to the LO file will only contain those source lines that have certain predefined character strings in them. You can also specify your own strings so that subsequent listings will also include source lines that have those strings in them. With LST, there is no predefined list, but you must specify one or more strings. All subsequent listings of source files to the LO file will then contain only those source lines that have the user-defined strings in them.

Because these commands have an effect on all other SED commands that generate listings of source files to the LO file, they will have an effect on the output of a LIST command.

For more information on modifying output listings, refer to the individual command descriptions in Chapter 3. LNAME, LNS, and LNT are described together in Chapter 3. Figure 3-7 under that description shows the effect of an LNT command on a subsequent LIST command. Figure 3-10 under the LST command shows the effect of an LST command on a subsequent LIST command.

2.7 REMOVING AND RENAMING PROGRAMS

2.7.1 Removing Programs from a USL Directory

Once a program is cataloged in a USL, it remains there until the USL file is destroyed or until you specifically delete the program from the directory. To delete a cataloged program entry from a USL directory, use the following command:

```
REMOVE name
```

where name is the one- to eight-character name of the program you want to delete.

If you want to remove more than one program at a time and the program names have certain identical sequences of characters in them, you can use the RAE command. The name specified in an RAE command can have plus signs (called wildcard characters) in the positions that will match any character. For example:

```
RAE NEW+++
```

will remove all programs from the USL directory whose names are six characters in length and that begin with the three characters NEW.

2.7.2 Renaming Programs

There are two SED commands that perform renaming functions: RENAME and TWIN.

RENAME changes the name of a cataloged program to another name:

```
RENAME name1 name2
```

The source program that was previously cataloged as name1 must now be referenced by name2.

To assign an alias name to a cataloged program, use the TWIN command:

```
TWIN name1 name2
```

With this command, the program that was originally cataloged as name1 can be referenced as either name1 or name2.

2.8 FILE MARKS

Up to this point, it has been assumed that each source program on the SI file is separated by the default system end-of-file mark, which is two dollar signs ($$). However, in some cases you may have a source program (with one file mark at the end) that really contains several programs, subroutines, or groups of macro definitions, each of which is not separated by a file mark. Suppose you want to copy only part of a program (for example, several of the subroutines) onto a tape without copying the entire program. Or suppose the program is very large and you want a hard-copy listing of only part of it without listing the program in its entirety. In other cases you may have several source programs together in sequential format as one file and you want to separate them into individual source files. By using the MARK command, you can divide a program into any arbitrary units. For example, you may want to separate the main program from the subroutines, then separate the subroutines from each other, then separate the macro definitions from the subroutines, etc.

To divide a source program into other units, the program must contain occurrences of a string of unique identifying characters at the points where it is to be divided. These strings will then become pseudo file marks and will be processed by most SED commands in the same manner as the $$ mark would be processed.

As an example, suppose you have a file cataloged in your USL that contains FORTRAN source code and that file is set up in the following form (the ellipses indicate the FORTRAN code):

```
      PROGRAM  MAIN
        ...
      END
NEW   SUBR 1
        ...
      RETURN
NEW   SUBR 2
        ...
      RETURN
NEW   SUBR 3
        ...
      RETURN
NEW   SUBR 4
        ...
      RETURN
      END
      $$ (system default EOF mark)
```

This file contains a main program and four subroutines. Because the character string NEW is a label that appears at the beginning of each subroutine, it can be used as a pseudo file mark by specifying the following command in sequential mode:

```
MARK 'NEW'
```

Until you redefine the file mark to something else, SED will interpret any source line in a program that has the string NEW starting in column one as the first statement of a subprogram.

To list or copy selected portions of a source program that contains a mark string, you can position to the program on the SI file and then define the string to be used as the file mark. In sequential mode the LIST and COPY commands accept an integer parameter by which you can tell SED to copy or list programs from the SI file until "integer" file marks have been encountered. The string you previously specified in the MARK command will then be treated as a file mark each time it is encountered.

The MARK command also has an effect if you are recataloging a program in sequential mode. If you have defined your own file marks and then recatalog the program for which those marks are in effect, SED will only recatalog the program up to that file mark.

The default form of the MARK command is:

```
MARK '$$' '$$'
```

which assumes that each source file consists of exactly one program and that subdivision do not exist.

As shown by its default form, the MARK command can actually take two strings, although the second string is optional:

```
MARK (string1) [(string2)]
```

If neither string is $$ or END, then the first string defines the file mark to divide a single source file into separate programs. The second string optionally defines another file mark to further subdivide those programs into subprograms, subroutines, macros, or any other unit that you choose.

There are special cases if either string is $$ or END. If either string is $$, SED interprets it as the system file mark (i.e., the end of a program). If either string is END, SED considers the mark to be a FORTRAN END statement. Any END statement within a source program will then be treated as the last source line of a program or subprogram. If the strings are anything other than $$ or END, SED assumes that any source line that contains the string starting in column one is the first line of a program or subprogram.

If either string1 or string2 is END, the other string (if present) must be $$. If string1 is $$, then string2 (if present) must be $$ or END. Therefore, the following combinations are the only valid ones that can be used if either string is $$ or END:

```
MARK 'END'
MARK 'END '$$'
MARK '$$'
MARK '$$' 'END'
MARK '$$' '$$'  (this is the default)
```

2.8.1 File Mark Output Control

SED normally writes a file mark to the SO file whenever it reads a file mark on the SI file. If you want to combine several source files that are separated by file marks into one sequential file, use the NOFILE command to prevent file marks from being written to the SO file:

```
NOFILE
```

To go back to the automatic writing of file marks on the SO file, use the FILE command:

```
FILE
```

When you first invoke SED, the FILE command is in effect by default.

2.9 COPYING SOURCE PROGRAMS

There are two SED commands that are used to perform copy functions: COPY and TRANSFER. COPY can be used to copy programs from the SI or WRK file to the SO file and also to copy an entire directory from disc to tape or from tape to disc. The records being copied can be compressed or uncompressed and can also be written to the SO file in either format. If the NOCOMPRESS command is in effect, the records are copied to SO in uncompressed format. COMPRESS is the default mode when you first invoke SED.

TRANSFER is used to copy entire programs from either the WRK file or the SI file to the SO file in compressed format. TRANSFER is not affected by the setting of various other SED commands.

The rest of the information in the following sections describes how to use these two commands to copy programs to the SO file.

2.9.1 Copy to SO File

In random mode the COPY command is used to copy a selected number of records from the WRK file to the SO file or to copy all of the records on the WRK file to the SO file. When COPY is used in random mode, SED automatically exits random mode and returns to sequential mode.

In sequential mode COPY can be used to copy records from the current position of the SI file to the SO file until a file mark is encountered, or to copy one or more entire source programs from the SI file to the SO file. File marks encountered on the SI file are written to the SO file unless the NOFILE command is in effect. This feature is useful if you want to want to combine several programs into one source file. When you first invoke SED, the FILE command is in effect, so if you do not want file marks to be copied, set NOFILE.

Because COPY in sequential mode reads from the current position of SI file, you must first position this file to the appropriate program if the records or programs you want to copy are not in the current position.

A special case of the COPY command is when you are editing a program in sequential mode and have not made changes all the way through to the last line in the program. In this case, you must use the COPY command to get the remaining original records from the SI file to the SO file before you recatalog the program.

For example, if the program you are editing has 45 lines and the last change you made was to line 30, you would have to copy lines 31 through 45 to the SO file. By using the COPY command after changing line 30, SED would copy records from the current position of the SI file (line 31) to the SO file until an EOF mark is encountered (after line 45). If you do not copy the rest of the program before recataloging it, those remaining lines will be lost. The use of the COPY command in this situation was discussed in Section 2.4.1, "Editing in Random and Sequential Modes."

To copy an entire source program in random mode from the WRK file to the current position of the SO file, use COPY without any parameters:

```
COPY
```

To copy an entire source program in sequential mode from the SI file to the current position of the SO file, first position to the desired program and then use COPY without any parameters as shown above. SED will copy the program until a file mark is encountered, also coping the file mark unless NOFILE has been specified.

In sequential mode an integer parameter can be specified with COPY to copy more than one program at a time:

```
COPY integer
```

SED will copy programs from the current position of the SI file until "integer" file marks have been encountered. A double file mark terminates the copy even if the specified number of programs have not been copied.

In random mode the integer parameter has a different meaning. It specifies the number of source lines to copy from the current position of the WRK file to the current position of the SO file. A file mark terminates the copy even if the specified number of lines have not been copied.

If you have divided your source programs into subprograms by defining other file marks with the MARK command, this will have an effect on subsequent COPY commands because those marks will be treated as file marks. If you are using other file marks and you also use COPY with an integer parameter, a system file mark ($$) will terminate the copy even if the specified number of lines, programs, or subprograms have not been copied.

2.9.2 Transfer to the SO File

The TRANSFER command is also used to copy source programs from the SI file (in sequential mode) or the WRK file (in random mode) to the current position of the SO file. The differences between TRANSFER and COPY are that TRANSFER is not affected by the COMPRESS, NOCOMPRESS, MARK, SEQUENCE, EXSEQUENCE, and UNSEQUENCE commands and in random mode, TRANSFER can only be used to copy the entire WRK file, not selected lines from the WRK file.

With TRANSFER, source programs are always copied to the SO file in compressed format, and the physical end-of-file ($$) is always used as the file mark.

The syntax of the TRANSFER command in random mode is:

```
TRANSFER
```

which copies the entire WRK file to the current position of the SO file in compressed format.

The syntax of the TRANSFER command in sequential mode is:

```
TRANSFER [integer]
```

The optional integer parameter specifies the number of programs to copy from the current position of the SI file (e.g., the number of file marks to bypass before terminating the transfer). A double EOF mark terminates the transfer even if the specified number of programs have not been copied. If you use TRANSFER in sequential mode without specifying an integer, only one program is copied.

2.9.3 Copying a Directory to Disc or Tape

One form of the COPY command, COPY ALL, can be used to copy all of the source programs in a USL from the SI file to the SO file. The requirements to use COPY ALL are:

- The SO file can be either a disc file with a directory or a magnetic tape with or without a directory.
- The SI file can be either a disc file or a magnetic tape, but must contain a directory.

Based on these requirements, COPY ALL can be used to copy a USL directory and all of the programs cataloged in the USL from disc to tape or from tape to disc, making it useful if you plan on transporting programs from one system to another.

The syntax of the COPY command to copy a USL file to a secondary storage device is:

```
COPY ALL [NOREWIND]
```

The optional NOREWIND parameter prevents the SI or SO file, if either is a tape, from being rewound. It is useful if you want to store more than one USL on the same tape. By default SI is rewound prior to the copy and SO is rewound after the copy. If both SI and SO are assigned to tapes, then you must explicitly position them and specify NOREWIND.

If the SI file is a magnetic tape that already has more than one USL directory on it, the following command can be used to position SI to the "integer" USL directory from its current position before doing a COPY ALL:

```
LEOF integer
```

If a USL directory is being copied to a disc, you must assign SO to a disc file and use the COPY command with the USL parameter:

```
COPY USL
```

SED will then initialize the disc with a directory size that is large enough to hold all of the programs that are being copied from the SI file.

NOTE: Once a directory is copied to a tape, it cannot be modified; that is, you cannot add programs to or delete programs from the directory on the tape, but you can search the directory for program names.

2.10 USING MACROS

A macro is a named sequence of commands that is executed when a call is made to the macro name. Once you define the macro name and the list of commands that make up the macro, you need only enter the macro name as a command to invoke its associated sequence of commands.

A macro definition conists of the following components:

1. A MACRO command followed by the macro name. For example:

   MACRO M1

   defines a macro named M1.

2. The list of commands to be executed when the macro is called.

3. An ENDMACRO command to terminate the macro definition.

Macro names can be one to three CAN-code characters in length. The macro name can also be the name of an existing SED command. This enables you to redefine a standard command to your own customized definition. If a macro name is the same as an existing command, SED first checks the macro list before checking its standard list of commands when that command is used.

Macro definitions can contain calls to other macros up to a nesting level of four, but a macro cannot call itself. The CONTINUE command can be used to force a macro to execute continually until the end-of-program (EOP) or beginning-of-program (BOP) is reached. If used, CONTINUE is placed just before the ENDMACRO command within the macro definition.

If a macro that contains a CONTINUE command is not constructed properly, SED may be put into a mode of continuous operation that cannot be terminated normally. If this happens, you can abnormally terminate the execution of the macro by activating the Operator Communication (OC) task and entering:

```
/R ABT
```

This will discontinue the current operation and you can then continue to enter other SED commands. Refer to the CONTINUE command description in Chapter 3 for more information.

Macro definitions can be removed by entering:

```
MACRO name REMOVE
```

To list the contents of a specific macro definition to the CO file, enter:

```
MACRO name LIST
```

To list the contents of all currently defined macro definitions to the CO file, enter:

```
MACRO
```

The number of macros defined at any one time is installation-dependent based on the amount of available memory space. This attribute can be changed by the system manager.

2.11 INSERT FILES

When you use an ADD or REPLACE command, by default SED expects to read the new records that are being added or the replacement records from the CI file. Normally CI is assigned to your terminal so that you can enter the new lines from the terminal. However, if the AO option is set, then SED expects to read the records from the AI file instead of from the CI file.

Another alternative is to read the input records from another logical file (e.g., from a scratch file or a cataloged program in your USL). If the input records are on a logical file other than CI or AI, you can use the INSERT command to force SED to read from the specified logical file for an ADD or REPLACE command:

```
INSERT file
```

The file parameter gives the name of the logical file on which the input source records or program is located. That logical file must first be positioned to the appropriate source file or location of the input records before you use the ADD or REPLACE command. Then when SED encounters an ADD or REPLACE command, it will read the records from the specified logical file until a file mark is encountered. A second INSERT command is needed after the ADD or REPLACE command to redirect command input back to either CI (if the AO option was not set) or AI (if the AO option was set).

The following sequence of commands shows the use of INSERT and ADD commands in random mode. In this example, the input records to be added are contained in the source file NEWDATA, which is a cataloged program in the USL. The records from NEWDATA are to be added after line 15 of the source file TESTER, which is the current program on the WRK file. Note that after the records are added, a second INSERT command is needed to redirect command input back to CI.

```
ASSIGN SI USL
POSITION TESTER
RANDOM
INSERT SI
POSITION NEWDATA
ADD 15
INSERT CI
...you can then contine to edit TESTER in random mode...
RECATALOG TESTER
```

The insert file can also contain other SED commands. However, the first SED command after the last input record must contain a terminator character to indicate the end of the addition or replacement records. By default the terminator character is a period in column one, just as would be used after an ADD or REPLACE command if you were entering the input records from the terminal.

For an example of inserting a file in sequential mode, refer to the INSERT command description in Chapter 3.

2.12 LABELS AND SEQUENCE NUMBERS

When the LIST command is used, SED lists source files statement-by-statement with sequential line numbers assigned to each statement. These numbers are followed by two blank spaces and the actual 80-byte record. The FORTRAN compiler and MODCOMP assemblers read 80-byte records but only process the information contained in the first 72 bytes. Bytes 73 through 80 are not processed but do appear as part of the source listing generated by SED.

In some cases you may want a source file listing to also contain labels and sequence numbers starting in column 73. This can help you to identify source records when you edit a program or if you want to make references to particular statements in the source program. The SEQUENCE, EXSEQUENCE, UNSEQUENCE, and NOSEQUENCE commands are used to control label and sequence number insertion in source files.

If the SEQUENCE command is used, SED places a four-digit sequence number in columns 77 through 80 of each source line. An optional four-character label can also be specified, in which case that string is placed in columns 73 through 76 of each source line. If a label is not specified in the SEQUENCE command, the data in columns 73 through 76 remains unchanged. The syntax of the SEQUENCE command is:

```
SEQUENCE [label]
```

The EXSEQUENCE command is an augment of the SEQUENCE command that can only be used after a SEQUENCE command has already been specified. If the SEQUENCE command specified a label and then EXSEQUENCE is used, the label is truncated to three characters and placed in columns 73 through 75 of each source line. Five-digit sequence numbers are then placed in columns 76 through 80 of each line. The syntax of the EXSEQUENCE command is:

```
EXSEQUENCE [NORESET]
```

The NORESET option prevents sequence numbers from being reset when SED encounters a file mark on the SI file. This feature is useful if you want to list several files together with continuous sequence numbers. If NORESET is not specified, the five-digit sequence number is reset to zero when an end-of-file mark is encountered.

To insert blank spaces in columns 73 through 80 instead of labels and sequence numbers, use the UNSEQUENCE command:

```
UNSEQUENCE
```

If you have used both SEQUENCE and UNSEQUENCE commands in the same SED session, the execution of UNSEQUENCE precedes the execution of SEQUENCE.

To stops any label writing and sequencing that was started by a SEQUENCE, EXSEQUENCE, or UNSEQUENCE command, use the NOSEQUENCE command:

```
NOSEQUENCE
```

NOSEQUENCE is the default mode when you first invoke SED (that is, no labels or sequence numbers appear on source file listings).

2.12.1 Protecting Label and Sequence Number Fields

The LIMIT command can be used to protect the last part of every source line from being overwritten by a CHANGE, MODIFY, OVERLAY, IST, or MST command. This is useful if you want to reserve those columns for inserting labels and sequence numbers as explained in the previous section.

If you want to use the LIMIT command, you must also set the LIMIT option in the current SED session:

```
OPTION LIMIT
```

and then use the LIMIT command:

```
LIMIT [integer]
```

If an integer is not specified, columns 73 through 80 will be protected. This means that the data in those columns cannot be overwritten by any command except SEQUENCE, EXSEQUENCE, or UNSEQUENCE. To protect a range of columns other than 73 through 80, specify the last unprotected column number as the integer parameter. For example:

```
LIMIT 44
```

protects columns 45 through 80 (i.e., column 44 is the last unprotected column).

2.12.2 Checking Sequence Numbers

The CHECK command can be used to check the sequence numbers in columns 77 through 80 of each source line on the SI file for ascending order. If the value of the sequence number in a source line is less than the value of the sequence number from the previous line, SED will report the error with the message SEQUENCE CHECK ERROR.

If you did not specify the HOLD option when you first invoked SED to hold after an error occurs, you should follow one of the four steps listed below after a SEQUENCE CHECK ERROR:

- Enter RSM to continue reading and checking sequence numbers, using the value of the sequence number from the last source line that was in sequence.
- Enter RSM LOW to continue reading and checking sequence numbers, resetting the value of the sequence counter to zero.
- Enter RSM NO to continue reading records without checking sequence numbers.
- Enter any other command to abort the current command and proceed. In this case, you should reassign the SI file.

If SED does enter a hold state, activate the Operator Communication (OC) task, enter the single character "/" and follow one of the first three steps above, using the character "R" in place of the characters "RSM."

SED continues to check sequence numbers after a CHECK command until you suppress it with the UNCHECK command:

UNCHECK

UNCHECK is the default mode when SED is first invoked.

NOTE: The CHECK command does **not** cause SED to examine the sequence numbers of records from the CI or AI file.

2.12.3 Resetting Sequence Numbers

SED uses sequence numbers as the input index to control many editing functions. In sequential mode, SED resets the input index when:

- SED is initially loaded
- A file mark is encountered on the SI file
- Any ASSIGN, REWIND, AVFILE, BKFILE, or WEOF command that operates on the input file SI is encountered.

In random mode the sequence number is the position of the current line on the WRK file. SED resets sequence numbers when new records are added or existing records are replaced or deleted.

The sequence number on updated source programs (i.e., a newly modified file) is the output index which appears on the new updated source listing. SED resets the output index when:

- SED is initially loaded
- A file mark is written on the SO file
- Any ASSIGN, REWIND, AVFILE, BKFILE, or WEOF command that operates on the output file SO is encountered.

2.13 SOURCE PROGRAM VERIFICATION

SED can also be used to verify the data integrity of source programs whose records are in compressed format. The source records to be verified must have been writtten by SED or the MAX IV Source Update processor.

An error in the checksum value of a compressed record causes SED to return a CHECKSUM ERROR message. If you did not enter the HOLD option when you first invoked SED to hold after an error occurs, follow one of the four steps listed below after a CHECKSUM ERROR:

- Enter RSM to continue reading records and checking for data integrity.
- Enter RSM NO to continue reading records without checking for data integrity (until an end-of-file mark is encountered).

- Enter RSM STOP to stop reading records and begin processing the next SED command. In this case you should reassign the source input (SI) file.

- Enter any other command to abort the current command and proceed. In this case you should also reassign the source input (SI) file.

If SED does enter a hold state, activate the Operator Communication (OC) task, enter the single character "/" and follow one of the first three steps, using the character "R" in place of the characters "RSM."

2.14 DIRECTORY MAINTENANCE

2.14.1 Listing the Contents of a Directory

The LDIRECTORY command lists all of the program names that are cataloged as entries in a USL directory and also gives statistical information about space utilization on the directory. The syntax of the LDIRECTORY command is:

```
LDIRECTORY
```

Figure 2-1 on the next page shows the output format of a directory listing when the USL is assigned to a disc partition. If the USL is assigned to a tape, only the names of the programs and the number of entries cataloged are reported by the LDIRECTORY command.

The names under the PROGRAM column are the names used when the programs were cataloged. The information displayed in the IDENTIFICATION column varies depending on what options were set when each program was cataloged. If the PGM option was set, then this column displays the name found on the first PGM or PROGRAM statement within the source file. If the PGM option was set and the program did not contain a PGM or PROGRAM statement, then this field is blank. If the DATE option was set, then this is the date and time that the program was cataloged or recataloged. If an identification string was specified on the CATALOG or RECATALOG command, then this field contains that string and the PGM and DATE options (whichever was set) are ignored.

In the statistical information, the number of directory entries used includes an entry for each program, a beginning directory entry, an ending directory entry, an entry for each directory sector, and an entry for each attached file. (The format of directory entries on disc are illustrated in Appendix B.) The number of unused directory entries is the number available before another directory sector would need to be added. The largest number of contiguous unused sectors is the size of the largest program that can currently be cataloged in the USL.

The NEEDS TO BE COMPRESSED message tells you that there are unused sectors scattered throughout the USL file. When this happens, the numbers in the START and END columns will not be continuous. Note the gaps in the numbers where one program ends and another begins between sectors 124 - 129, 144 - 147, and 148 - 173. This means that you should squeeze the USL so that the unused sectors will be placed contiguously at the end of the USL. Squeezing a USL is explained in Section 2.14.2.

```
*G.Ø  MODCOMP SOURCE EDITOR       DATE  Ø9/27/85  1Ø:41:58        PAGE   1
 DIRECTORY LISTING OF $FM$     I.D.= DEBBIE       LAST MODIFIED   Ø9/26/85

PROGRAM       IDENTIFICATION       FILE    START     END    LENGTH

ASMEXAMP          ENDEXA           USL       4       15       12
EXAMP2            ENDEXA           USL      16       26       11
MULTMAC     Ø2/Ø6/85   15:5Ø       USL      27       28        2
SINE        Ø2/Ø7/85   15:Ø6       USL      29       3Ø        2
NEWEQU      Ø2/Ø8/85   12:44       USL      31       33        3
MODFILE     Ø2/Ø8/85   12:54       USL      34       35        2
IFAR        Ø2/Ø8/85   14:19       USL      36       37        2
LASTONE     Ø2/11/85   1Ø:Ø7       USL      38       41        4
SLIST       Ø2/2Ø/85   14:Ø2       USL      42       43        2
EQUATES     Ø2/22/85   15:Ø6       USL      44       46        3
ORIGIN      Ø2/22/85   15:11       USL      47       51        5
MACNOGO     Ø3/26/85   17:1Ø       USL      52       53        2
XREFEX      Ø4/18/85   16:42       USL      54       56        3
NULLARG     Ø5/Ø1/85   16:39       USL      57       58        2
ARGPAR      Ø5/Ø1/85   16:4Ø       USL      59       61        3
MODES       Ø5/Ø1/85   16:42       USL      62       64        3
IFNOT       Ø5/Ø1/85   16:43       USL      65       67        3
COMPGO      Ø5/Ø1/85   16:46       USL      68       7Ø        3
DEFIELD     Ø5/Ø1/85   16:47       USL      71       73        3
COMMON      Ø5/Ø1/85   16:48       USL      74       78        5
BOUND       Ø5/Ø1/85   16:49       USL      79       81        3
ATRIB       Ø5/Ø1/85   16:5Ø       USL      82       86        5
COS         Ø5/Ø1/85   16:54       USL      87       88        2
TAN         Ø5/Ø1/85   16:56       USL      89       9Ø        2
AØSINE      Ø5/Ø1/85   17:58       USL      91       92        2
SUBS        Ø5/Ø1/85   18:ØØ       USL      93       95        3
CON32       Ø5/Ø1/85   18:Ø1       USL      96       98        3
COUNT       Ø5/15/85   13:3Ø       USL      99      1Ø1        3
NEWTEMP     Ø5/15/85   13:32       USL     1Ø2      1Ø3        2
LISTER      Ø5/15/85   13:33       USL     1Ø4      1Ø6        3
EXAMP       Ø6/12/85   14:43       USL     1Ø7      116       1Ø
IFINT       Ø6/24/85   17:21       USL     117      118        2
MODF        Ø6/24/85   17:31       USL     119      12Ø        2
MARKS       Ø7/Ø8/85   16:41       USL     121      122        2
TESTER      Ø7/1Ø/85   16:2Ø       USL     123      124        2
NEWTEST                            USL     129      13Ø        2
TEMPEX                             USL     131      132        2
TEST                               USL     133      136        4
LINETEST                           USL     137      138        2
JUNK            TEST               USL     139      14Ø        2
TRATEST         TESTPGM            USL     141      142        2
ARGMIS      Ø9/13/85   18:3Ø       USL     143      144        2
MOD                                USL     147      148        2
SPOT        Ø9/26/85   17:16       USL     173      174        2
MARKTEST    Ø9/26/85   2Ø:2Ø       USL     175      176        2
NUMBER OF ENTRIES CATALOGED =     45
NUMBER OF DIRECTORY ENTRIES USED =      52
NUMBER OF UNUSED DIRECTORY ENTRIES =       4
LARGEST NUMBER CONTIGUOUS UNUSED SECTORS =    3Ø23
TOTAL NUMBER OF UNUSED SECTORS =    3Ø53
TOTAL NUMBER OF SECTORS USED =     147
ATTACHED FILES :  USL
NEEDS TO BE COMPRESSED
```

Figure 2-3. Format of a Directory Listing

Other parameters available with the LDIRECTORY command are:

NOR[EWIND]	If the USL is assigned to a magnetic tape, this parameter prevents it from being rewound before listing the directory.
TES[T]	Checks the integrity of each program as the directory is listed. This paramter can only be used if the USL is assigned to a disc.
ARC	Used to list the directory of a USL that is in saved format. Refer to Section 2.14.4 for an explanation of saved format).

For more information on any of these parameters, refer to the description of the LDIRECTORY command in Chapter 3.

The LDS command can be used to report only the statistical information that is normally reported at the end of an LDIRECTORY listing. The LDE command can be used to report all of the information normally reported by LDIRECTORY except the statistical information.

2.14.2 Squeezing a Directory

If the NEEDS TO BE COMPRESSED message appears at the end of a directory listing, it means that there are unused sectors scattered throughout the USL file. When this happens, you should use the SQUEEZE command to compress all directory sectors and source files on the USL. This will make the scattered sectors available so that other source files can be cataloged in the USL. The syntax of the SQUEEZE command is:

```
SQUEEZE [NOTEST]
```

SED saves the USL and any attached files to the SO file and then restores them to the USL file when the squeeze is completed. By default, SED tests the saved and restored files for data integrity. If NOTEST is specified, SED does not check for data integrity. A directory listing is automatically written to the LO file after the squeeze is completed. By examining the columns marked START and END in this listing, you will be able to see that the existing programs are located on contiguous sectors throughout the USL.

2.14.3 Attaching and Detaching Space from Other Partitions

You can make the space of a particular disc partition available to hold source programs that are cataloged in a USL file by attaching the file:

```
ATTACH file
```

The logical file to be attached must be a disc file but it cannot be a MAX IV or MAX 32 File Manager file.

When future programs are cataloged in the USL, the space on the attached file is available when SED attempts to find space to hold the source programs. Before a program can be cataloged on any file other than USL, at least one ATTACH command must be used since the directory was initialized. To keep the source files separate from the directory, you can detach the USL file by using:

```
DETACH USL
```

To take away the space from the USL for a file that was previously attached, also use the DETACH command, using the same file name that was used to attach the file:

```
DETACH file
```

An error occurs if any cataloged programs still reside on the file being detached. You must first delete all of the cataloged programs that reside on the attached file by using the REMOVE command (as discussed in Section 2.7.1).

2.14.4 Saving and Restoring USL Files

The SAVE command can be used to copy all source files in a USL, the USL directory, and any files that are attached to the USL to another device. The USL is copied to the SO file in a special saved format that can only be read by the RESTORE and TEST commands and by the LDIRECTORY command when used with the ARC parameter. The format of saved USL files is illustrated in Appendix B.

The syntax of the SAVE command is:

```
SAVE
```

To restore a previously saved USL, use the RESTORE command. The current file assignments for the USL and any attached files must be the same as when the SAVE command was used to save the USL file. The format of the RESTORE command is:

```
RESTORE [integer]
```

This command copies the USL from the SI file back to the USL file. If the SI file contains more than one USL, the integer parameter can be used to indicate which saved USL file is to be restored (number of double file marks to be advanced minus one).

To test a saved USL file for data integrity, use the TEST command:

```
TEST
```

The USL to be tested must be on the SI file.

CHAPTER 3
SED COMMANDS

3.1 SPECIAL SYMBOLS AND NOTATIONS USED IN THIS CHAPTER

Character Strings

Some SED commands described in this chapter require you to supply a parameter that consists of one or more characters. This parameter is generally used when you want to search through a source file to find the occurrence of a particular string or to modify one or more characters within a source line. The syntax used to indicate a character string parameter for which you must supply a value has the following form:

```
. (string)
```

Strings must be enclosed in delimiters, such as parentheses as in the preceding example. Valid delimiters for character strings can be any of the following:

```
() [] <> {} '' "" :: || -- .. //
```

The character used as a delimiter cannot appear as an actual character in the delimited string.

The period (.) or backslash (/) should not be used to delimit numerical strings.

Symbols

Double quotes (") are used to delimit a parameter or set of symbols in text when that set of symbols has just appeared in an example or is being discussed.

Brackets [ ] indicate that the enclosed item is optional. It is mainly used in describing the syntax prototype for a command and to show acceptable abbreviations for a command. For example:

```
[.] DUP[LICATE] [SLR1 [SLR2]]
```

means that the DUPLICATE command can be abbreviated as DUP and that you have a choice of supplying an SLR1 parameter, both SLR1 and SLR2 parameters, or none at all. Default values are supplied if optional parameters are not explicitly specified.

When a command has more than one valid abbreviation and the use of brackets to denote all possible abbreviations would be confusing, each abbreviation is explicitly spelled out. For example:

```
BACKWARD [integer]
BAC
B
```

indicates that acceptable abbreviations for the BACKWARD command are BAC and B.

If parameters within brackets are stacked, they are optional. However, if you do specify a parameter, you can only choose one. For example:

```
                [integer]
COPY[Y]         [ALL [NOREWIND]]
                [USL]
```

means that with the COPY command, you can specify an integer, the keyword ALL (with or without the keyword NOREWIND), the keyword USL, or no parameters at all.

When parameters are enclosed in braces, you must choose one of the given values. For example:

```
                { integer [integer] ... }
CLE[ARTAB]      { ALL                   }
```

means that valid parameters for the CLEARTAB command are a series of one or more integers or the keyword ALL.

Point

A point (.) is a special character that can be used by itself or as the first character of a command. It is the terminator character that signals the end of text input for an ADD, REPLACE, NPU, IPU, NRE, or IRE command. If the point is entered on a line by itself, SED recognizes that you are finished entering input and is ready to accept another command. If the point is the first character of a command, SED recognizes that you are finished entering input and also processes that command. For example, the following two command sequences are equivalent:

```
ADD 2
...enter new source lines...
.
LINE
...output from the LINE command would appear here...
```

```
ADD 2
...enter new source lines...
.LINE
...output from the LINE command would appear here...
```

Thus, all commands in this chapter have as their first character an optional point that can be used in this way. When the point and command are specified on the same line, note that the command must immediately follow the point (without any intervening blanks).

3.3 SED/JOB CONTROL COMMANDS

There are some SED commands that have the same name and perform the same function as the corresponding Job Control commands. This manual assumes that you are familiar with these commands. Only a brief description of the use and syntax of these commands are presented below. For more information, refer to the MAX IV/MAX 32 Nonresident Job Control Programmer's Reference Manual as listed in the Preface.

[.]ACT[ION] message
: Writes a message to the CO file and causes SED to enter a hold state. Execution must be resumed through the Operator Communication (OC) task.

[.]ASS[IGN] file file [file file] ...
: Assigns a logical file to a device or another logical file. When the SI file is addressed, SED resets the input sequence number to zero. When the SO file is addressed, SED resets the output sequence number to zero. The WRK file cannot be the logical file named in the assignment.

[.]AVF[ILE] file [integer]
: Advances a logical file a specified number of end-of-file marks. When the SI file is addressed, SED resets the input sequence number to zero. When the SO file is addressed, SED resets the output sequence number to zero. If an integer is not specified, the logical file is backspaced by one file mark.

[.]AVR[ECORD] file [integer]
: Advances a logical file a specified number of records. If an integer is not specified, the logical file is advanced by one record.

[.]BKF[ILE] file [integer]
: Backspaces a logical file a specified number of end-of-file marks. When the SI file is addressed, SED resets the input sequence number to zero. When the SO file is addressed, SED resets the output sequence number to zero. If an integer is not specified, the logical file is backspaced by one file mark.

[.]BKR[ECORD] file [integer]
: Backspaces a logical file a specified number of records. If an integer is not specified, the logical file is backspaced by one record.

$NOP [comment]
: Null command that is ignored by SED. It is typically used to insert comments into a job stream.

[.]NOT[E] message
: Writes a message to the CO file. SED does not enter a hold state after the message, but continues to read the next command.

[.]REW[IND] file [file] ...
: Rewinds a logical file to its beginning position. When the SI file is addressed, SED resets the input sequence number to zero. When the SO file is addressed, SED resets the output sequence number to zero. The WRK file cannot be rewound.

[.]WEO[F] file [file] ...

Writes an end-of-file mark on the specified logical file. When the SI file is addressed, SED resets the input sequence number to zero. When the SO file is addressed, SED resets the output sequence number to zero. You cannot write an end-of-file to the WRK file with the WEOF command.

3.3 SED COMMANDS

The remainder of this chapter contains individual descriptions of each SED command. Commands are listed in alphabetical order and include a functional description, a line illustrating the proper syntax of the command, and where necessary for clarification, usage examples and sample output.

Add New Records to a Source Program

The ADD command is used to add new lines of text to a source program. By default the records to be inserted are read from the CI file. Normally this file is assigned to your terminal so that you can enter the new input records from your terminal. If the AO option is set, the records to be inserted are read from the AI file. If the INSERT command is used, the records to be inserted are read from the logical file named on that command.

In random mode SED reads the records from the appropriate file (as described above) and inserts them into the WRK file after the line specified in the ADD command. The current line pointer is then set to the last line that was added.

In sequential mode SED copies the original source records from the current position of the SI file to the SO file, up to the line after which the new records are to be added. SED then copies the source additions from the appropriate file (as described above) until a point (.) in column one or an end-of-file mark ($$) is encountered.

A point in column one, either by itself or immediately followed by a SED command on the same line, is the default character used to terminate input of records after the ADD and REPLACE commands. This can be changed to some other character with the DEFINE command (using the TERMINATOR parameter of that command).

If the records to be inserted already exist on another logical file (i.e., you are not entering them from the terminal), you can use the INSERT command before the ADD command to name the logical file where the records are located. SED will then read from the insert file until a file mark is encountered, inserting the records after the line specified in the ADD command. If the insert file also contains SED commands, the first character of the first command must be a terminator character to indicate the end of the addition records. This application of ADD and INSERT is useful to insert the records of a program that is cataloged in a USL into another cataloged program. An example illustrating this use of ADD in conjunction with INSERT is shown in the description of the INSERT command in this chapter and also in Section 2.11 in Chapter 2.

Also refer to Section 2.5.2 in Chapter 2 for additional information about the ADD command.

SYNTAX

[.] A[DD] [SLR]

[SLR] (OPTIONAL) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. It specifies the line after which new records are to be added. If not specified, new records are added after the current line. To add new lines before line one, use a zero for the SLR parameter (e.g., ADD 0).

EXAMPLES

.ADD 20

Adds new records after line 20.

ADD 'ROUTINE'+3

Allows you to insert new lines three lines after the first line encountered that contains the string ROUTINE.

A,1

Allows you to insert new lines after line 1.

.A |XYZ|

Allows you to insert new lines after the first line encountered that contains the string XYZ.

Attach File to USL Directory

The ATTACH command is used to make the disc space on the specified logical file available for cataloging programs in the USL directory. The file being attached must be assigned to a disc partition.

SYNTAX

[.] ATT[ACH] file

file (REQUIRED) Parameter 1 is the name of the logical file to be attached. It cannot be the name of a MAX IV or MAX 32 File Manager file.

EXAMPLES

ATTACH SCA

ATT SCB

BACKWARD --

Backspace WRK or SI File by Records

The BACKWARD command backspaces a logical file by a specified number of records.

In random mode the BACKWARD command operates on the WRK file. In sequential mode it operates on the SI file and can only be used if the SI file is a disc or tape.

SYNTAX

[.] BACKWARD [integer]
BAC
B

[integer] (OPTIONAL) Parameter 1 is a positive integer specifying the number of source lines to be backspaced. If not specified, the appropriate file is backspaced by one source line.

EXAMPLES

BACKWARD 4

B 6

BAC 2

Copy All Entries

The CAE command is used to search the directory of the USL file for specific programs and copy the contents of those programs to the SO file. Wildcard characters can be used when specifying the program name so that any character in the position where a wildcard appears will match the specified program name. Wildcard characters are specified in the form of plus signs. This enables you to copy more than one program at a time to the SO file based on certain characters in the source program names.

SYNTAX

[.] CAE name

name (REQUIRED) Parameter 1 is a one- to eight-character program name with wildcard characters (plus signs) in the positions that will match any character.

EXAMPLES

CAE ++++OLD

Searches for all programs whose names are seven characters in length and that end with the three characters OLD. If any matching names are found, the contents of those programs will be copied to the SO file.

CAE F:+++

Searches for all programs whose names are five characters in length and that begin with the two characters F:. If any matching names are found, the contents of those programs will be copied to the SO file.

CAE M+++$

Searches for all programs whose names are five characters in length, begin with the character M, and end in a dollar sign. If any matching names are found, the contents of those programs will be copied to the SO file.

CATALOG

Catalog Programs in a USL

The CATALOG command copies the contents of a source program to the USL file and creates a directory entry for it. The program being cataloged cannot not be null.

In random mode SED copies the source file that is currently on the WRK file. In sequential mode SED copies the source file from the current position of the SI file until an end-of-file mark is encountered.

If * is used instead of a program name, the program is cataloged with the name that was used in the most recent POSITION command.

If the DATE option is in effect, the date and time of the catalog are recorded in the USL directory. If the PGM option is in effect, the name found on the first PGM or PROGRAM statement in the program is recorded instead of the date and time of the catalog. If an identification is specified, it is recorded and the DATE and PGM options have no effect. The date and time, PGM or PROGRAM name, or identification is displayed under the IDENTIFICATION column of a directory listing. By default the PGM option is in effect when SED is first invoked.

SYNTAX

[.] CAT[ALOG] $\left\{\begin{matrix}\text{name} \\ * \end{matrix}\right\}$ [id]

Parameter 1 is required and can be either a name or an asterisk:

name the one- to eight-character name under which to catalog the source program.

* to catalog the program with the name used in the most recent POSITION command.

[id] (OPTIONAL) Parameter 2 is a one- to eight-character name that identifies the source program. If used, this information is recorded in the USL directory and the DATE or PGM option (whichever is set) is ignored. The identification will be displayed under the IDENTIFICATION column in a directory listing.

EXAMPLES

CATALOG FILE4
Catalogs a program named FILE4.

CAT,SOURCE
Catalogs a program named SOURCE.

CAT * NEWDIAG
Catalogs the current program with the name used in the most recent POSITION command and also records the identification NEWDIAG in the USL directory.

Change Characters within a Source Record

The CHANGE command is used to replace a specific character string with another string. The change can be made one or more times per source line on the current line only or for a specific number of lines, starting with the current line.

The integer1 parameter is the number of times per source line that the change is to be made. If not specified, the change is only made once per source line. The integer2 parameter is the number of source lines to search for the string. If not specified, the change is only made on the current line as many times as specified by the integer1 parameter.

If the string to be replaced and the replacement string are not specified, SED uses the character strings from the most recent CHANGE command. Any values from the integer1 and integer2 parameters are not carried over.

The search for the characters to be changed begins on the current line and continues for "integer2" lines. The current line pointer is then set to the last line that was searched, even if the change is not made on that line.

If the REVISION command is in effect, the revision string will be inserted into every source line affected by a CHANGE command. If the NOQUICK option or command is in effect, the line or lines that are changed will be displayed after they are changed.

To reverse the effect of the most recent CHANGE command, use the UNDO command. The line pointer must be positioned on the line on which the change was originally made.

Other commands that can be used to modify source lines are OVERLAY, MODIFY, IST, and MST. Refer to Section 2.5.6 in Chapter 2 or the descriptions of those commands in this chapter for more information.

SYNTAX

[.] CHANGE [(string1) (string2)] [integer1 [integer2]]
CHA
C

[string1] (OPTIONAL) Parameter 1 is the character string to be replaced. It must be enclosed in any of the valid string delimiters that were listed in Section 3.1. If not specified, SED uses the value of string1 from the most recent CHANGE command.

[string2] (OPTIONAL) Parameter 2 is the character string that is to replace string1. It must be enclosed in any of the valid string delimiters that were listed in Section 3.1. If not specified, SED uses the value of string2 from the most recent CHANGE command.

[integer1] (OPTIONAL) Parameter 3 is a positive integer specifying the number of times per source line that the change is to be made. If not specified, the change is only made once per source line.

[integer2] (OPTIONAL) Parameter 4 is a positive integer specifying the number of source lines to search for the string that is being changed. "integer2 + 1" lines are actually searched because the search begins at the current line and continues for integer2 lines. If not specified, the change is only made on the current line as many times as specified in the integer1 parameter.

EXAMPLES

CHANGE 'ALPHA','BETA',1,50

Changes the first occurrence of ALPHA to BETA in each souce line, beginning at the current line and continuing for 50 lines.

C :B:,:C:

Changes the first occurrence of the character B to C on the current line only.

C :B:C:

Same as the preceding example except that the commas can be omitted because colons are used as delimiters.

CHA /SUM/SUMS/

Changes the first occurrence of the string SUM to SUMS on the current line only.

CHECK

Check Sequence Numbers

The CHECK command verifies that sequence numbers in columns 77 through 80 of the SI file are in ascending order. Sequence numbers are generated by the SEQUENCE or EXSEQUENCE commands.

SYNTAX

[.] CHE[CK]

CLEARTAB

Clear Tabs

The CLEARTAB command clears one or more of the current tab settings. The ALL parameter is used to remove all tabs.

To set new tabs, use the SETTAB command. To display the current tab settings, use the PTA or SCALE command.

For more information on the use of tabs to format input data, refer to Section 2.2.4 in Chapter 2.

SYNTAX

[.] CLEARTAB
CLEAR
CLE
{ integer [integer] ... | ALL }

EXAMPLES

CLE 7,21,36,73
Clears the tabs that are set at columns 7, 21, 36, and 73.

CLEAR ALL
Clears all of the current tab settings.

COMPRESS
CONTINUE

COMPRESS

Write to SO File in Compressed Format

The COMPRESS command causes all subsequent records that are written to the SO file to be in compressed format. The COMPRESS command itself does not write records, but has an effect on all other SED commands that do write records to the SO file. COMPRESS is the default mode when SED is first invoked.

To force records that are written to the SO file to be in uncompressed format, use the NOCOMPRESS command. COMPRESS would then be used to return to writing compressed records.

Note that if records are copied to the SO file with the TRANSFER command, the NOCOMPRESS command, if in effect, is ignored. TRANSFER always copies records in compressed format, although the records being copied can be compressed or uncompressed. To copy records to the SO file in uncompressed format, use the NOCOMPRESS and COPY commands as shown in the example under the NOCOMPRESS command.

SYNTAX

[.] COM[PRESS]

CONTINUE

Continue Macro Processing

The CONTINUE command is used to cause a macro to execute continuously until the beginning-of-program (BOP) or end-of-program (EOP) is reached. If used, the CONTINUE command is placed after the sequence of commands that make up a macro definition and immediately before the ENDMACRO command as follows:

```
MACRO name
...sequence of commands for the macro definition...
CONTINUE
ENDMACRO
```

SYNTAX

[.] CON[TINUE]

Copy Source Programs to SO File

The COPY command copies records to the SO file. In random mode the records are copied from the WRK file. In sequential mode the records are copied from the SI file.

The integer parameter has different meanings depending on whether you are doing the copy in random or sequential mode.

In random mode it specifies the number of source lines to copy from the WRK file, starting at the current line. SED remains in random mode after copying the specified number of lines. The position of the current line is not changed when COPY is used in this way. If the integer parameter is not specified, SED copies the contents of the entire WRK file to the SO file and then exits random mode.

In sequential mode the integer parameter specifies the number of programs to copy from the current position of the SI file to the SO file. If not specified, SED copies one program, beginning at the current line on the SI file, until a file mark is encountered.

If the MARK command has been used, SED copies the appropriate number of programs, subprograms, or records, using the user-defined mark as a file mark. A $$ mark terminates the copy even if the specified number of programs, subprograms, or records have not yet been copied.

If the MARK command has not been used, SED copies the appropriate number of programs or records to the SO file, using $$ as the file mark. A double file mark terminates the copy even if the specified number of programs or records have not yet been copied.

In sequential mode file marks on the SI file are copied over to the SO file unless the NOFILE command is in effect.

COPY ALL is used to copy every source file on the SI file to the SO file. In this case, the SI file must contain a directory. If the SO file is a disc file, the new programs are added to the existing directory. If the SO file is a magnetic tape, the USL directory is also copied to the tape. If the source USL is empty, nothing is written to the tape.

COPY ALL NOREWIND is used to prevent the tape from being rewound if either the SI or SO file is assigned to a tape. If both SO and SI are assigned to tape, then you must explicitly position the tape and specify NOREWIND. If NOREWIND is not specified, SI is automatically rewound prior to the copy and SO is automatically rewound after the copy.

If the USL parameter is specified, SED first initializes the SO file and builds enough directory entries at the beginning of it to hold all the programs on the SI file plus eight more for expansion. SED then performs a COPY ALL.

SYNTAX

```
                [integer]
[.] COP[Y]      [ALL [NOREWIND]]
                [USL]
```

Parameter 1 is optional. If specified, it can be one of the following values or keywords:

integer a positive integer specifying the number of source lines to copy from the WRK file (in random mode), or the number of programs or subprograms to copy from the SI file (in sequential mode). If not specified, SED copies either the entire WRK file (in random mode) or one program or subprogram beginning at the current position of the SI file (in sequential mode).

ALL specifies to copy all source programs from the SI file to the SO file. The USL file is not squeezed before the programs are copied, so any "holes" due to fragmented space on the USL are also copied over to the SO file. Use the USL parameter or squeeze the USL before the copy for efficient use of file space.

ALL NOR[EWIND]

prevents the SI or SO file from being rewound if either is assigned to a tape. If NOREWIND is not specified with ALL, the SI file is rewound before the copy and the SO file is rewound after the copy.

USL creates a directory on the SO file and then performs a COPY ALL. The USL is squeezed before the copy so that any "holes" due to fragmented space on the USL are not copied over to the SO file.

EXAMPLES

COPY 6

If this command is used in random mode, it would copy six records from the WRK file to the SO file. If this command is used in sequential mode, it would copy six programs or subprograms from the current position of the SI file to the SO file.

.COP

If this command is used in random mode, it would copy all of the records from the WRK file to the current position of the SO file. If this command is used in sequential mode, it would copy the program from the current position of the line pointer on the SI file to the SO file.

COP ALL

Copies every program from the USL file to the SO file. If there are any "holes" in the USL due to fragmented space, they will also be copied.

Redefine Special Characters

The DEFINE command is used to display the defaults or change one or more of the defaults for the following characters:

- The tab character used when creating new source files, adding or replacing records with the ADD and REPLACE commands, or modifying characters with the OVERLAY command (default = CONTROL-I keys).
- The character used to insert a blank space as an actual character with the MODIFY and OVERLAY commands (default = &).
- The character used to terminate the addition or replacement of records with an ADD, REPLACE, NPU, IPU, NRE, or IRE command (default = .).
- The character used to delete characters with the MODIFY command (default = _).

If no parameters are specified with the DEFINE command, SED displays the current definitions in effect for these characters.

SYNTAX

[.] DEF[INE] keyword { (character) / #integer }

keyword — Parameter 1 is a keyword indicating the function (as described above) whose character you want to redefine. The keyword can be one of the following: TAB, SPACE, TERMINATOR, or UNDERSCORE. SPACE, TERMINATOR, and UNDERSCORE can be abbreviated as SPA, UND, and TER, respectively.

Parameter 2 can only be specified if a keyword was specified in Parameter 1. Parameter 2 can be one of the following:

(character) — A single character that is enclosed in any of the valid string delimiters that were listed in Section 3.1. The corresponding function specified for Parameter 1 will then be defined as this character.

#integer — A one- or two-digit unsigned hexadecimal integer that represents the desired character's ASCII code. The # must be specified exactly as shown.

EXAMPLES

DEF SPACE "/"

Subsequent use of / when using the MODIFY and OVERLAY commands would produce a blank space.

DEFINE TAB #5E

The ASCII code for hexadecimal 5E is an uparrow (^). Subsequent use of ^ when adding or replacing source records, or when modifying characters using the OVERLAY command, would produce a tab. This would be the same as specifying DEFINE TAB :^:

DELETE

Delete Records from a Source Program

The DELETE command is used to delete one or more lines from a source program. In random mode SED deletes the specified line(s) from the WRK file. In sequential mode SED copies the original source records from the current position of the SI file to the current position of the SO file, up to the first source statement to be deleted. If a second parameter is specified to delete a range of lines, SED then deletes source lines up to and including the line specified by the SLR2 parameter.

To delete the current line only, use DELETE without any parameters.

SYNTAX

[.] DELETE [SLR1 [SLR2]]
DEL
D

[SLR1] (OPTIONAL) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. It specifies the first line in a range or the only line to be deleted. If not specified, SED deletes the current line.

[SLR2] (OPTIONAL) Parameter 2 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. It specifies the last line in the range of lines to be deleted. If not specified, SED deletes the single line indicated by SLR1 or the current line in neither SLR parameter is specified.

EXAMPLES

DEL 'RTN','RTN'+6
Deletes the range of lines from the first source line encountered that contains the string RTN through the next six lines.

D 4 7
Deletes lines 4 through 7.

DELETE 21|ROUTINE|-1,|XYZ|
Deletes source lines from one line before the line that contains the string ROUTINE starting in column 21 through the first line encountered that contains the string XYZ.

DEL
Deletes the current line.

Detach a Previously Attached File

The DETACH command removes the space for a previously attached file from the USL directory. The space on the attached file is no longer available for cataloging programs in a USL. Before you use the DETACH command, you must first delete all programs that are both cataloged in the USL directory and that reside on the specified file. This can be done by using the REMOVE command.

SYNTAX

[.] DET[ACH] file

file (REQUIRED) Parameter 1 is the name of the logical file that was previously used in an ATTACH command.

EXAMPLES

DET SCA

DETACH WRK

DISPLAY

Display Current Source Program a Screenful at a Time

The DISPLAY command displays text from a source program to the CO file a screenful at a time. Parameters can be used to display the current, previous, or next screen.

If the TWO option is in effect, a screenful is 10 lines; if the ONE option is in effect, a screenful is 20 lines. Option TWO is the default mode when SED is first invoked.

After each display the current line pointer is set to the first line that is displayed on the terminal screen.

SYNTAX

[.] DIS[PLAY] [keyword]

Parameter 1 is optional. If specified, it can be one of the following keywords:

C[URRENT] Displays the current screen. This is the default condition if a parameter is not specified.

B[ACK] Displays the previous screen.

F[ORWARD] Displays the next screen.

Define String

The DST command is used to define a character string that can later be inserted into one or more lines with an IST command. The DST command defines the string while the IST command identifies the particular source line into which it should be inserted.

The string defined in a DST command also has an effect on the use of an MST command used without a parameter. MST is used to delete a specified character string from the current line. However, if MST is used without specifying a character string, SED will use the string from the most recent MST command. If an MST command has not been used or if a DST command has been used since the last MST, SED will then use the string from the most recent DST command.

The string defined in a DST command is remembered until you exit SED or until you redefine it with another DST command or use an MST command with a string. The string used in the most recent MST command always becomes the currently defined string, just as if you had defined it with the DST command.

For more information on the use of DST, MST, and IST, refer to Section 2.5.6.4 in Chapter 2 or the command descriptions in this chapter.

SYNTAX

[.] DST (string)

(string) (REQUIRED) Parameter 1 is the character string you want to insert with an IST command or delete from the current line with an MST command. The string must be enclosed in any of the valid string delimiters that were listed in Section 3.1.

EXAMPLES

```
DST "ABCD"
IST 10
```

Displays line 10 so that you can enter an uparrow (^) to indicate where to insert the string ABCD.

```
DST :,X7:
MST
```

Deletes the string ",X7" from the current line.

```
DST /ABCD/
MST /END/
```

Removes the string END from the current line and also makes that string the currently defined string for the DST command. Subsequent use of an IST command would then assume that you want to insert the string END into the specified line.

DUPLICATE

Duplicate Source Lines

The DUPLICATE command is used to reproduce one or more lines within a source program and place them after the current line. The lines to be duplicated can be located before or after the current line unless the SLR1 and/or SLR2 parameters are character string references. In that case the lines to be duplicated must be located on or after the current line.

After the lines are duplicated, they are not deleted from their original location. To move one or more lines and delete them from their original location, use the MOVE command. MOVE can only be used in random mode.

In random mode SED resets the line pointer to the last line that was duplicated in the new location. In sequential mode SED locates the specified line or range of lines on the SI file and writes them to the SO file. In sequential mode the current line is not changed by a DUPLICATE command.

SYNTAX

[.] DUP[LICATE] [SLR1 [SLR2]]

where SLR1 and SLR2 are source line references as previously described in Sections 2.4.2.1 through 2.4.2.3. To duplicate the current line, omit the SLR1 and SLR2 parameters. To duplicate one line, specify only the SLR1 parameter. To duplicate a range of lines, specify the first line as SLR1 and the last line in the range as SLR2.

EXAMPLES

DUPLICATE 'SUBR1','SUBR1'+8

Duplicates the range of lines starting with the first line that contains the string SUBR1 through the next eight lines, and places them after the current line.

DUP C,C+12

Duplicates the range of lines from the current line through the next 12 lines and places them after the current line.

DUP 2,8

Duplicates lines two through eight and places them after the current line.

.DUP 1<ABC>,1<ABC>+9

Duplicates the range of lines from the first line that contains the string ABC starting in column one through the next nine lines, and places them after the current line.

DUP

Duplicates the current line.

ENDMACRO

End Macro Definition

The ENDMACRO command is used to terminate a macro definition that was started by a MACRO command.

Refer to the descriptions of the MACRO or CONTINUE commands in this chapter for a usage example of ENDMACRO.

SYNTAX

[.] END[MACRO]

EXIT

Exit SED

The EXIT command is used to terminate a SED editing session and return to Job Control. It can be used from either random or sequential mode. However, if you want to exit random mode but still remain in SED, use the XRN or RANDOM EXIT command.

SYNTAX

[.] EXIT
EXI
E

EXSEQUENCE

Generate Five-Digit Sequence Numbers

The EXSEQUENCE command is an augment of the SEQUENCE command that can only be used after a SEQUENCE command has been used. EXSEQUENCE causes the sequence numbers that were started by SEQUENCE to be five digits instead of four. The five-digit sequence numbers will be placed in columns 76 through 80.

If a four-character label was specified in the SEQUENCE command and EXSEQUENCE is also used, the label is truncated to three characters and placed in columns 73 through 75. If EXSEQUENCE is used and a label was not specified with SEQUENCE, then the data in columns 73 through 75 remain unchanged.

The NORESET option prevents sequence numbers from being reset when an end-of-file is encountered. This is useful if you want to obtain listings of more than one source program at a time with continuous sequence numbers. The effect of NORESET can be stopped by using another EXSEQUENCE command without any parameters or by using the NOSEQUENCE command. However, NOSEQUENCE will also stop the generation of sequence numbers and labels.

For more information on the use of sequence numbers, refer to Section 2.12 in Chapter 2 or the descriptions of the SEQUENCE, NOSEQUENCE, and UNSEQUENCE commands in this chapter.

SYNTAX

```
[.] EXS[EQUENCE] [NORESET]
                 [NOR]
```

FILE

Write EOF Marks to SO File

The FILE command causes SED to write an end-of-file record to the SO file whenever it reads an end-of-file mark on the SI file.

FILE is the default mode when SED is first invoked. This command is generally used to reverse the effect of a previous NOFILE command.

SYNTAX

[.] FIL[E]

FORWARD

Advance Line Pointer by Records

The FORWARD command is used to access a specific line by moving the pointer forward by a specified number of lines. In random mode the FORWARD command operates on the WRK file. In sequential mode it operates on the SI file and the records that are bypassed are **not copied** to the SO file.

If the NOQUICK command or option is in effect, the current line is displayed on the CO file after a FORWARD command.

SYNTAX

[.] FORWARD [integer]
FOR
F

[integer] (OPTIONAL) Parameter 1 is the number of records to move the line pointer. If not specified, the line pointer is advanced one line forward.

GOTO

Move Line Pointer

The GOTO command has the same effect as entering a line number as a command itself; that is, it moves the line pointer from its current position to the specified line within the source file. That line then becomes the new current line.

In random mode you can go to a line that is before or after the current line. In sequential mode you can only go to a line that is located after the current line because SED copies the lines that are bypassed from the current position of the SI file to the SO file.

If the SLR parameter is a character string reference and column limits have been imposed by a previous LIMIT command, those limits are in effect when SED searches for the character string.

To initiate a backward search in random mode using a character string reference, precede the string delimiter with a minus sign as shown in the last two examples below. If the SLR parameter is a line number and that line is somewhere before the current line, you do not have to use the minus sign. For example, GOTO 4 will move the pointer to line 4 even if the current line is line 20.

After a GOTO using a character string reference, you can then use GOTO without a parameter to go to the next or previous occurrence of that string. The search starts at the current line plus one if the most recent GOTO used a forward character string reference. If the SLR parameter from the most recent GOTO used a backward character string reference, the search starts at the current line minus one and goes backward.

Note that when a character string is used for the SLR parameter (either forward or backward), the search for that string always starts at the current line. Therefore if the string appears on the current line, the line pointer will remain on that line. Entering another GOTO without an SLR parameter will then move the pointer to the next or previous occurrence of that string as described above. When the string can no longer be found, SED will return the message SOURCE LINE NOT FOUND.

When GOTO is used in sequential mode, SED writes the original source lines that are bypassed from the current position of the SI file to the current position of the SO file. A listing of the source lines that were copied to the SO file are also listed on the LO file.

If the NOQUICK option or command is in effect, SED displays the current line on the CO file after a GOTO command.

In random mode you can also move the line pointer forward one line at a time by simply pressing the carriage return.

SYNTAX

```
[.] GOTO [[-]SLR]
    GOT
    G
```

[SLR] (OPTIONAL) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. It references the line that you want to go to. If not specified, SED uses the SLR value from the previous GOTO or SKIPTO command, whichever was used most recently. If the search is to be a backward search using a character string reference, the minus sign must precede the string delimiter. If a line number is used, the minus sign is not required (see the examples below). Backward searches can only be done in random mode.

EXAMPLES

G 144

Moves the pointer to line 144.

.GOTO C-12

Move the pointer to the line that is 12 lines before the current line. This would only be valid in random mode.

GOTO 1{ROUTINE}

Moves the pointer to the first line encountered that contains the string ROUTINE starting in column one. If it is found on the current line, the pointer will remain on the current line. Entering another GOTO without a parameter would then move the pointer to the next occurrence of the string ROUTINE in column one.

82

Moves the pointer to line 82.

G -/HEX/

Searches backward for the line that contains the string HEX. If that string appears on the current line, the pointer would remain on the current line. Entering another GOTO without a parameter would then move the pointer to the previous occurrence of the string. This would only be valid in random mode.

G -/ALPHA/-1

Searches backward for the line that contains the string ALPHA and then sets the pointer to one line before that. If the string appears on the current line, the pointer will be set to one line before the current line. Entering another GOTO without a parameter would cause SED to search for the previous occurrence of the string and set the pointer to one line before that. This would only be valid in random mode.

HITHER (Random Mode Only)

Save Current Position in WRK File

The HITHER command can only be used in random mode. It causes SED to save the current position of the line pointer in the WRK file so that it can be restored when a YON command is issued. For example, if the current line is line 4 and you enter:

```
HITHER
```

SED will remember line 4 as the current line. You can then continue to work in random mode, making changes to other lines in the the program, thereby changing the position of the line pointer. At any time you can enter:

```
YON
```

and the current line will then become line 4.

Note that if you add, delete, or replace any lines in the file, the YON command will restore the "new" line 4, not the line that was originally line 4.

HITHER saves the current line number even if you exit random mode, position to another program on the SI file, and then enter random mode again. The only two ways to clear the line number saved by HITHER is to enter another HITHER command from random mode or exit SED.

SYNTAX

```
[.] HITHER
    HIT
    H
```

Write Buffer to Logical File

The HOME command writes a data buffer that is partially filled with compressed source records to the specified file (normally the SO file).

SYNTAX

[.] HOM[E] file

file (REQUIRED) Parameter 1 is the name of the logical file to which the compressed records are to be written.

INITIALIZE

Initialize USL Directory

The INITIALIZE command is used to create a directory on the USL file. Once a directory is created, source programs can then be cataloged on the USL file. In order to use this command, the USL file must be assigned to a disc partition. The USL file is automatically attached when INITIALIZE is used.

If the HOLD option is in effect when the INITIALIZE command is used, SED will enter a hold state and write a warning message that you have requested an initialization. If you resume execution (through the Operator Communication task or Terminal Monitor Program), SED builds a new directory, even if a directory already exists on the USL file. **When you initialize a directory, all previous data on the USL file will be lost.**

SYNTAX

[.] INITIALIZE [integer] [id]
INIT
INI

[integer] (OPTIONAL) Parameter 1 is the number of entries (source programs) that will be cataloged in the directory. If not specified, SED builds the directory with one sector.

[id] (OPTIONAL) Parameter 2 is a one- to eight-character USL identifier name. If specified, it is recorded in the USL directory and displayed in the I.D. field at the top of a directory listing. See Figure 3-3 under the LDIRECTORY command for a sample directory listing showing the I.D. field.

Read Text Input from Alternate File

The INSERT command specifies the name of an alternate logical file from which the text input for source statement additions and replacements should be read. If an INSERT command is used, SED reads from the specified file instead of from the CI or AI file when an ADD or REPLACE command is used. A terminator character or a file mark signals the end of the input data on the insert file. By default the terminator character is a period in column one (refer to the DEFINE command for more information on terminator characters).

After specifying the INSERT and then the ADD or REPLACE commands, you must reassign input back to either CI or AI by using the INSERT command again (see the following examples). For more information on insert files, refer to Section 2.11 in Chapter 2.

SYNTAX

[.] INS[ERT] file

file (REQUIRED) Parameter 1 is the name of the logical file from which SED should read the input records after an ADD or REPLACE command.

EXAMPLES

```
ASSIGN SI USL
POSITION LETTER
RANDOM
INSERT SI
POSITION TEXT
ADD 20
INSERT CI
```

This sequence of commands inserts the source records contained in the file named TEXT after line 20 of the file named LETTER, which is currently on the WRK file. The file TEXT must also be a cataloged program in the USL directory since SI is assigned to the USL.

```
POSITION DATA
GOTO 95
ASSIGN X SCA SO SCB
INSERT X
ADD
INSERT CI
COPY
ASSIGN SI SCB
RECATALOG DATA
```

This example merges text from a scratch file on logical file X after line 95 of the file DATA, which is on the SI file. Because this example is done in sequential mode, the GOTO command first copies lines 1 through 95 of DATA to the SO file. The ADD command then inserts the records from the scratch file after line 95, the current position of the SI file. After redirecting input back to the CI file, the COPY command then copies lines 96 through the end of DATA (from the SI file) to the SO file. After recataloging DATA, that file now contains the inserted records from the scratch file.

Insert String

The IST command is used in conjunction with a DST command to insert a character string into one or more source lines. The string to be inserted must have already been defined in a previous DST command. When you are ready to insert the string, use the IST command to identify the particular line on which to insert the string. This feature is useful if you want to insert the same string into more than one source line because you only have to define the string once. You can also insert the same string into more than one source file because the string defined in a DST command is remembered until you exit SED or until you redefine the string in another DST command or use the MST command with a string.

After an IST command, SED displays the specified source line on the CO file. To insert the currently defined string, enter an uparrow (^) below the character before which the string is to be inserted. To insert the string into the current line, omit the SLR parameter from the IST command. If the NOQUICK command or option is in effect, SED will display the line after the string is inserted.

As mentioned above, the string defined in a DST command will be changed if you use an MST command with a character string. MST is used to remove a specified string from the current line. The string used in an MST command then becomes the currently defined string, just as if you had defined it with the DST command. If you then use an IST command without redefining the string with DST, SED assumes that you want to insert the string that was used in the most recent MST command.

SYNTAX

[.] IST [SLR]

where SLR is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. If not specified, SED displays the current line so that the string from the previous DST or MST command can be inserted (whichever was used most recently).

EXAMPLES

.IST 10

Displays line 10 so that the string specified in the most recent DST command can be inserted. If you used an MST command with a string after the most recent DST command, SED will display line 10 so that the string from that MST command can be inserted.

IST C+5

Displays the line that is five lines after the current line so that the string specified in the most recent DST command can be inserted. If you used an MST command with a string after the most recent DST command, SED will display the line that is five lines after the current line so that the string from that MST command can be inserted.

DST /END/
.IST 6"MAC"

Displays the first line encountered that contains the string MAC starting in column six so that the string END can be inserted.

Limit Character String Searches to Specific Columns

The LABEL command is a useful adjunct to any command that is used to search for or locate a character string. By issuing the LABEL command, you can set up a particular range of columns that SED will examine for the character string. Columns that are outside the imposed limits will not be examined when you use any SED command to subsequently search for a particular character string.

If the integer1 and integer2 parameters are specified, subsequent searches for character strings will be limited to that range of columns. If the integer1 and integer3 parameters are specified, subsequent searches will be limited to "integer3" columns beginning at column "integer1."

If you have previously used a LABEL command and want subsequent searches to span across all columns of all source lines, enter the LABEL command without any parameters. This will override any values previously specified in the integer1, integer2, and integer3 parameters of the LABEL command.

Any limits imposed by a LABEL command also affect the search for a character string when the SLR parameter of any SED command is a character string reference. For example, if you specify:

```
LABEL 1-20
GOTO /HEX/
```

and the string HEX only appears after column 20, it will not be found by the GOTO command because the LABEL command restricts the search for all character strings to columns one through 20.

SYNTAX

[.] LAB[EL] integer1-integer2

or

[.] LAB[EL] integer1 integer3

where integer1, integer2, and integer3 are positive integers that represent column positions. Note that if the first form is used, the dash (-) between integer1 and integer2 is required.

EXAMPLES

LABEL 30-50
: Limits all subsequent searches for character strings to columns 30 through 50.

LABEL 30 20
: Limits all subsequent searches for character strings to columns 30 through 50 (starting at column 30 and continuing for 20 columns).

LAE

List All Entries

The LAE command is used to search the directory of the USL file for specific programs and list the contents of those programs or just the program names. Wildcard characters can be used when specifying the program name so that any character in the position where a wildcard appears will match the specified program name. Wildcard characters are specified in the form of plus signs. This enables you to search for or list programs even if you do not know the full program name, such as listing the names of all programs in the directory that begin or end with certain sequences of characters. For example:

```
LAE A+++++++
```

will list the names of all programs in the USL directory that are eight characters in length and that begin with the character A.

Use the LIST parameter to list the contents of the programs when any matching names are found. If LIST is not specified, SED reports only the names of the matching programs.

All output from an LAE command is listed to the LO file. If the LIST parameter is used, any restrictions imposed by a previous LNAME, LNS, LNT, LST, or XLN command will affect the content of the source file listings (see Figure 3-8 under the LNAME/LNS/LNT command).

SYNTAX

[.] LAE name [LIST]

name (REQUIRED) Parameter 1 is the name of the cataloged program for which to search, using wildcard characters (plus signs) in the positions that will match any character.

LIS[T] (OPTIONAL) Parameter 2 specifies to also list the contents of the programs when any matching names are found.

Figure 3-1 shows the output format of an LAE command that requests a list of program names that are six characters in length and that begin with the three characters ARG.

```
LAE ARG+++

*G.Ø  MODCOMP SOURCE EDITOR       DATE  Ø9/24/85  13:53:24          PAGE   4
 DIRECTORY LISTING OF $FM$     I.D.= DEBBIE       LAST MODIFIED   Ø9/24/85


PROGRAM       IDENTIFICATION      FILE   START     END    LENGTH

ARGPAR      Ø5/Ø1/85   16:4Ø     USL       59      61         3
ARGMIS      Ø9/13/85   18:3Ø     USL      143     144         2
```

Figure 3-1. Output Format of an LAE Command

List All Source Records

The LALL command is used to override a previous LST, LNT, LNS, LNAME, or XLN command. These commands are used to affect the content of source file listings that are generated to the LO file by only listing source lines that contain certain predefined and/or user-defined strings.

LALL is the default mode when SED is first invoked. See Figures 3-5, 3-6, 3-7, and 3-8 under the LNAME/LNS/LNT description for examples of listings with and without LALL in effect.

SYNTAX

[.] LAL[L]

List Directory Entries

The LDE command lists the names of all source programs that are cataloged on a USL file. The directory listing is written to the LO file. Optional parameters are provided to test the source programs for data integrity and to prevent the tape from being rewound if the USL file is assigned to a tape.

The directory listing from LDE is the same as that generated by an LDIRECTORY command except that with LDE, the statistical information at the bottom of the listing is not included. Figure 3-2 shows a directory listing generated by an LDE command.

SYNTAX

```
[.] LDE   [TEST]
          [NOREWIND]
```

Parameter 1 is optional. If specified, it can be either of the following keywords:

TES[T] Can only be used if the USL file is assigned to a disc. If specified, SED checks each source file in the USL for data integrity.

NOR[EWIND] Can only be used if the USL file is assigned to a magnetic tape. It prevents the tape from being rewound before listing the directory. If not specified, the tape is rewound.

Figure 3-2 is a sample of the output from an LDE command when the USL file is assigned to a disc. Source programs that have a date and time entry in the IDENTIFICATION column were cataloged with the DATE option in effect. Entries where the IDENTIFICATION field is blank were cataloged with the PGM option, but there was no PGM or PROGRAM statement within the source program. Entries that contain a name in the IDENTIFICATION column were either cataloged with the PGM option, and this was the name found on the first PGM or PROGRAM statement, or else the identification parameter of the CATALOG or RECATALOG command was specified when the program was cataloged or recataloged.

```
LDE

*G.Ø  MODCOMP SOURCE EDITOR        DATE  Ø9/24/85  13:53:17          PAGE   3
 DIRECTORY LISTING OF $FM$     I.D.= DEBBIE       LAST MODIFIED   Ø9/24/85

PROGRAM      IDENTIFICATION      FILE   START     END    LENGTH

ASMEXAMP        ENDEXA           USL      4       15       12
EXAMP2          ENDEXA           USL     16       26       11
MULTMAC   Ø2/Ø6/85   15:5Ø       USL     27       28        2
SINE      Ø2/Ø7/85   15:Ø6       USL     29       3Ø        2
NEWEQU    Ø2/Ø8/85   12:44       USL     31       33        3
MODFILE   Ø2/Ø8/85   12:54       USL     34       35        2
IFAR      Ø2/Ø8/85   14:19       USL     36       37        2
LASTONE   Ø2/11/85   1Ø:Ø7       USL     38       41        4
SLIST     Ø2/2Ø/85   14:Ø2       USL     42       43        2
EQUATES   Ø2/22/85   15:Ø6       USL     44       46        3
ORIGIN    Ø2/22/85   15:11       USL     47       51        5
MACNOGO   Ø3/26/85   17:1Ø       USL     52       53        2
XREFEX    Ø4/18/85   16:42       USL     54       56        3
NULLARG   Ø5/Ø1/85   16:39       USL     57       58        2
ARGPAR    Ø5/Ø1/85   16:4Ø       USL     59       61        3
MODES     Ø5/Ø1/85   16:42       USL     62       64        3
IFNOT     Ø5/Ø1/85   16:43       USL     65       67        3
COMPGO    Ø5/Ø1/85   16:46       USL     68       7Ø        3
DEFIELD   Ø5/Ø1/85   16:47       USL     71       73        3
COMMON    Ø5/Ø1/85   16:48       USL     74       78        5
BOUND     Ø5/Ø1/85   16:49       USL     79       81        3
ATRIB     Ø5/Ø1/85   16:5Ø       USL     82       86        5
COS       Ø5/Ø1/85   16:54       USL     87       88        2
TAN       Ø5/Ø1/85   16:56       USL     89       9Ø        2
AØSINE    Ø5/Ø1/85   17:58       USL     91       92        2
SUBS      Ø5/Ø1/85   18:ØØ       USL     93       95        3
CON32     Ø5/Ø1/85   18:Ø1       USL     96       98        3
COUNT     Ø5/15/85   13:3Ø       USL     99      1Ø1        3
NEWTEMP   Ø5/15/85   13:32       USL    1Ø2      1Ø3        2
LISTER    Ø5/15/85   13:33       USL    1Ø4      1Ø6        3
EXAMP     Ø6/12/85   14:43       USL    1Ø7      116       1Ø
IFINT     Ø6/24/85   17:21       USL    117      118        2
MODF      Ø6/24/85   17:31       USL    119      12Ø        2
MARKS     Ø7/Ø8/85   16:41       USL    121      122        2
TESTER    Ø7/1Ø/85   16:2Ø       USL    123      124        2
ABC                               USL    125      128        4
NEWTEST                           USL    129      13Ø        2
TEMPEX                            USL    131      132        2
TEST                              USL    133      136        4
LINETEST                          USL    137      138        2
JUNK            TEST              USL    139      14Ø        2
TRATEST         TESTPGM           USL    141      142        2
ARGMIS    Ø9/13/85   18:3Ø       USL    143      144        2
SPOT            TESTDATA          USL    145      146        2
MOD                               USL    147      148        2
```

Figure 3-2. LDE Directory Listing

LDIRECTORY —

List Directory

The LDIRECTORY command lists the names of all source files on the USL directory and also reports statistical information about use of directory sectors. If the USL is assigned to a disc file, the statistical information includes:

- Number of entries cataloged
- Number of directory entries used
- Number of unused directory entries
- Largest number of contiguous unused sectors
- Total number of unused sectors
- Total number of sectors used
- Attached files
- NEEDS TO BE COMPRESSED message (if unused space is fragmented)

Output from an LDIRECTORY command is directed to the LO file (see Figure 3-3 for a sample directory listing).

An LDIRECTORY command is automatically executed after squeezing a USL.

SYNTAX

```
                    [TEST]
[.] LDI[RECTORY]    [ARC [integer]]
                    [NOREWIND]
```

Parameter 1 is optional. If specified, it can be one of the following keywords:

TES[T] Can only be used if the USL file is assigned to a disc. If specified, SED checks each source file in the USL for data integrity.

ARC or ARC integer

Can only be used if the USL file is assigned to a file that is in saved format (produced by the SAVE command). An optional integer can be used with ARC if the USL contains more than one saved file. It specifies the number of double file marks minus one to advance before listing the saved file directory. If not specified, the default value is one.

NOR[EWIND] Can only be used if the USL file is assigned to a magnetic tape. It prevents the tape from being rewound before listing the directory. If not specified, the tape is rewound.

Figure 3-3 is a sample directory listing (see Figure 3-2 under the LDE command for an explanation of the entries in the IDENTIFICATION column). In Figure 3-3, the "NEEDS TO BE COMPRESSED" message appears at the end of the listing because directory sectors in the USL are fragmented. Note the gaps between sectors where one program ends and the next one starts in the START and END columns (e.g., between sectors 39 - 42, 68 - 71, 126 - 129, 130 - 133, etc.). When this USL is squeezed with the SQUEEZE command, the sectors in the BEGIN and END columns will be numbered contiguously.

```
LDI

*G.Ø  MODCOMP SOURCE EDITOR       DATE  Ø9/24/85  13:53:13          PAGE   1
 DIRECTORY LISTING OF $FM$     I.D.= DEBBIE       LAST MODIFIED   Ø9/24/85


 PROGRAM       IDENTIFICATION       FILE   START     END    LENGTH

 ASMEXAMP         ENDEXA            USL        4      15        12
 EXAMP2           ENDEXA            USL       16      26        11
 MULTMAC     Ø2/Ø6/85   15:5Ø       USL       27      28         2
 SINE        Ø2/Ø7/85   15:Ø6       USL       29      3Ø         2
 NEWEQU      Ø2/Ø8/85   12:44       USL       31      33         3
 MOD         Ø2/Ø8/85   12:52       USL       34      35         2
 MODFILE     Ø2/Ø8/85   12:54       USL       36      37         2
 IFAR        Ø2/Ø8/85   14:19       USL       38      39         2
 LASTONE     Ø2/11/85   1Ø:Ø7       USL       42      45         4
 SLIST       Ø2/2Ø/85   14:Ø2       USL       46      47         2
 EQUATES     Ø2/22/85   15:Ø6       USL       48      5Ø         3
 ORIGIN      Ø2/22/85   15:11       USL       51      55         5
 MACNOGO     Ø3/26/85   17:1Ø       USL       56      57         2
 XREFEX      Ø4/18/85   16:42       USL       58      6Ø         3
 NULLARG     Ø5/Ø1/85   16:39       USL       61      62         2
 ARGPAR      Ø5/Ø1/85   16:4Ø       USL       63      65         3
 MODES       Ø5/Ø1/85   16:42       USL       66      68         3
 IFNOT       Ø5/Ø1/85   16:43       USL       71      73         3
 COMPGO      Ø5/Ø1/85   16:46       USL       74      76         3
 DEFIELD     Ø5/Ø1/85   16:47       USL       77      79         3
 COMMON      Ø5/Ø1/85   16:48       USL       8Ø      84         5
 BOUND       Ø5/Ø1/85   16:49       USL       85      87         3
 ATRIB       Ø5/Ø1/85   16:5Ø       USL       88      92         5
 COS         Ø5/Ø1/85   16:54       USL       93      94         2
 TAN         Ø5/Ø1/85   16:56       USL       95      96         2
 AØSINE      Ø5/Ø1/85   17:58       USL       97      98         2
 SUBS        Ø5/Ø1/85   18:ØØ       USL       99     1Ø1         3
 CON32       Ø5/Ø1/85   18:Ø1       USL      1Ø2     1Ø4         3
 COUNT       Ø5/15/85   13:3Ø       USL      1Ø5     1Ø7         3
 NEWTEMP     Ø5/15/85   13:32       USL      1Ø8     1Ø9         2
 LISTER      Ø5/15/85   13:33       USL      11Ø     112         3
 EXAMP       Ø6/12/85   14:43       USL      113     122        1Ø
 IFINT       Ø6/24/85   17:21       USL      123     124         2
 MODF        Ø6/24/85   17:31       USL      125     126         2
 MARKS       Ø7/Ø8/85   16:41       USL      129     13Ø         2
 TESTER      Ø7/1Ø/85   16:2Ø       USL      133     134         2
 ABC                                USL      147     15Ø         4
 NEWTEST                            USL      151     152         2
 TEMPEX                             USL      153     154         2
 TEST                               USL      155     158         4
 LINETEST                           USL      159     16Ø         2
 JUNK             TEST              USL      163     164         2
 TRATEST          TESTPGM           USL      171     172         2
 ARGMIS      Ø9/13/85   18:3Ø       USL      175     176         2
 SPOT        Ø9/19/85   11:Ø3       USL      185     186         2
 NUMBER OF ENTRIES CATALOGED =      45
 NUMBER OF DIRECTORY ENTRIES USED =      52
 NUMBER OF UNUSED DIRECTORY ENTRIES =       4
 LARGEST NUMBER CONTIGUOUS UNUSED SECTORS =    3Ø13
 TOTAL NUMBER OF UNUSED SECTORS =    3Ø51
 TOTAL NUMBER OF SECTORS USED =     149
 ATTACHED FILES :  USL
 NEEDS TO BE COMPRESSED
```

Figure 3-3. LDIRECTORY Directory Listing

List Directory Statistics

The LDS command lists the same statistical information about the use of directory sectors that is reported at the end of an LDIRECTORY listing. For USL files that are assigned to disc, the statistical information includes the following:

- Number of entries cataloged
- Number of directory entries used
- Number of unused directory entries
- Largest number of contiguous unused sectors
- Total number of unused sectors
- Total number of sectors used
- Attached file names
- NEEDS TO BE COMPRESSED message (if unused space is fragmented)

If the USL file is assigned to a magnetic tape, LDS only reports the number of directory entries used.

Output from an LDS command is directed to the LO file. A sample of the output format is shown in Figure 3-4.

SYNTAX

[.] LDS [keyword]

Parameter 1 is optional. If specified, it can be one of the following keywords:

TES[T] can only be used if the USL file is assigned to a disc. If specified, SED checks each source file in the USL for data integrity.

NOR[EWIND] can only be used if the USL file is assigned to a magnetic tape. It prevents the tape from being rewound before listing the directory information. If not specified, the tape is rewound.

```
LDS

*G.Ø  MODCOMP SOURCE EDITOR       DATE  Ø9/24/85  13:53:15         PAGE   2
 DIRECTORY LISTING OF $FM$     I.D.= DEBBIE       LAST MODIFIED   Ø9/24/85


NUMBER OF ENTRIES CATALOGED =     45
NUMBER OF DIRECTORY ENTRIES USED =      52
NUMBER OF UNUSED DIRECTORY ENTRIES =       4
LARGEST NUMBER CONTIGUOUS UNUSED SECTORS =    3Ø51
TOTAL NUMBER OF UNUSED SECTORS =    3Ø51
TOTAL NUMBER OF SECTORS USED =     149
ATTACHED FILES :  USL
```

Figure 3-4. Output Format of an LDS Command

Advance SI File by Double File Marks

The LEOF command is used to advance the SI file by a specified number of double file marks. Double file marks exist at the end of each USL. LEOF is useful to position the SI file if it is a magnetic tape that contains multiple USLs.

SYNTAX

[.] LEO[F] [integer]

[integer] (OPTIONAL) Parameter 1 is the number of double file marks to advance the SI file. If not specified, the SI file is advanced one double file mark.

LIMIT

Limit Columns that Can Be Modified

The LIMIT command is used when the LIMIT option is set to prevent the last part of every line within a source file from being affected by a CHANGE, MODIFY, OVERLAY, IST, or MST command.

You must set the LIMIT option in order to use the LIMIT command. By default, the NOLIMIT option is in effect when you first invoke SED. If you do not explicitly set the LIMIT option, then the LIMIT command has no effect.

SYNTAX

[.] LIM[IT] [integer]

[integer] (OPTIONAL) Parameter 1 specifies the last column number that can be modified. This parameter must be in the range 2 through 80. If not specified and the LIMIT option is also set, columns 73 through will be protected.

EXAMPLES

LIMIT 44

Protects columns 45 through 80 from being modified.

As shown in the following example, LIMIT is useful to realign the left side of a line without affecting the right side. For example, suppose the source line is an Assembly language statement and appears as follows (where the numbers above the statement indicate the starting column positions of the text):

```
6           21          36
LDM,R6      LABEL       GET LABEL INFORMATION
```

Because column standards for MODCOMP Assembly language programs are 8, 21, and 36, the "LDM,R6" field needs to be shifted two places to the right without affecting the rest of the line. This can be accomplished by the following LIMIT and CHANGE commands:

```
OPTION LIMIT
LIMIT 20
CHA ::␣␣:
```

where ␣ represents a blank space. In this example, the limits in effect would protect columns 21 through 80. Since the first parameter of the CHANGE is null, this would insert two blank spaces at the beginning of the line, making "LDM,R6" start in column 8 without affecting columns 21 through 80.

Display Current Line Number

The LINENUMBER command displays the number of the current line on the CO file. The characters "BOP" are displayed if the pointer is positioned before the beginning of the program. The characters "EOP" are displayed if the pointer is positioned beyond the end of the program.

SYNTAX

[.] LIN[ENUMBER]

LIST

List Source Programs

The LIST command has two variations depending on whether you are operating in random or sequential mode.

In random mode LIST is used to list the entire contents of the WRK file or a specific number of lines from the WRK file. The integer parameter specifies the number of lines to list, beginning at the current line. If an integer is not specified, SED lists the contents of the WRK file from the beginning and sets the pointer to the end-of-program (EOP). The line pointer is not changed if you only list a specific number of lines through the integer parameter.

In sequential mode LIST can only be used to list entire source programs or subprograms from the SI file. In this case, the integer parameter specifies the number of programs to list, beginning at the current position of the SI file. If an integer is not specified, one program is listed and the pointer is set to the EOP. One program after another from the SI file can be listed by entering a succession of LIST commands without any parameters. Each listing terminates when SED encounters either the default EOF mark ($$) or another mark as defined in a previous MARK command.

The ALL parameter is used the same way for both random and sequential modes. If specified, SED lists all of the programs from the beginning of the USL file. **In this case only,** SED reads from the USL file.

The output from a LIST in both random and sequential modes is always directed to the LO file. To modify the output from any command that lists to the LO file by only listing source lines that contain certain predefined and/or user-defined character strings, use the LNT, LNAME, LNS, LST, or XLN commands. Refer to the descriptions of those commands in this chapter for more information and examples.

SYNTAX

```
[.] LIST      [integer]
    LIS       [ALL]
    L
```

Parameter 1 is optional. If specified, it can be one of the following:

[integer] In random mode, specifies the number of source lines to list from the WRK file. If not specified, the whole WRK file is listed. In sequential mode, specifies the number of programs or subprograms to list from the SI file. If not specified, the next program or subprogram on the SI file is listed.

[ALL] Lists all of the programs on the USL file.

Modify Output Listings

The LNAME, LNS, and LNT commands are all synonymous. They are used prior to any command that generates listings of source files to the LO file (e.g., LIST and LAE with the LIST parameter). If the LNAME, LNS, or LNT commands are used without parameters, all subsequent source listings on the LO file will only contain those lines with the following character sequences in them:

Special Identifier	Meanings and/or Position
((	Macro insert file identification, starting in column one
PGM name	Starting in other than column one
*CAT NUM	Where "*" is in column one and "CAT NUM" starts in any other column (used for MCS catalog numbers)
*CAT NO	Same as *CAT NUM
*CAT #	Same as *CAT NUM
CCAT NUM	Same as *CAT NUM
CCAT NO	Same as *CAT NUM
CCAT #	Same as *CAT NUM
END	Starting in other than column one
BLOCK DATA	Starting in other than column one
BLOCKDATA	Starting in other than column one
PROGRAM	Starting in other than column one
SUBROUTINE	Starting in other than column one
OVERLAY	Starting in other than column one
FUNCTION	Starting in other than column one
INTEGER FUNCTION	Starting in other than column one
LOGICAL FUNCTION	Starting in other than column one
REAL FUNCTION	Starting in other than column one
DOUBLE PRECISION FUNCTION	Starting in other than column one
COMPLEX FUNCTION	Starting in other than column one
TTL	Starting in other than column one

LNT, LNS, and LNAME can also take as parameters one or more user-defined strings. If specified, all subsequent source listings on the LO file will then contain those lines with the character sequences listed above and the lines with the user-defined strings.

To override the effect of a previous LNAME, LNS, or LNT command, use the LALL command. LALL is the default mode when SED is first invoked.

SYNTAX

[.] LNA[ME] [(string1) (string2)] ...

[.] LNS [(string1) (string2)] ...

[.] LNT [(string1) (string2)] ...

The optional character strings must each be enclosed in any of the valid string delimiters that were listed in Section 3.1. If no strings are specified, any command that lists source files to the LO file will only list the lines that contain any of the predefined character sequences.

Figures 3-5, 3-6, 3-7, and 3-8 show the effect of the LNT command on subsequent listings of source files. The first two figures are listings of files named ARGMIS (3-5) and ARGPAR (3-6) that were generated by a LIST command when LALL was in effect (the default mode when SED is first invoked). In Figure 3-7, an LNT command was used (LNS or LNAME could have been used instead of LNT and would have produced the same effect). Now note that the subsequent LIST command only listed those lines from ARGPAR that contained the character sequences from the list on the previous page. In Figure 3-8, an LAE command was used with the LIST parameter to list the contents of all source files whose names are six characters in length and that begin with the three characters ARG. Two matching program names were found: ARGPAR and ARGMIS. Because the LNT command was still in effect, note that the LIST command only listed the source lines from both programs that were found to contain any of the character sequences in the list on the previous page.

```
POS ARGMIS
LIST

 1  * ASSEMBLER CONDITIONAL DIRECTIVES
 2  *
 3  TEST   MAC
 4         IFP             %1,A
 5         DFC             5
 6  A      IFM             %1,DONE
 7         DFC             %1
 8  DONE   AOP
 9         EMP
10  *
11  * SOME CALLS TO THE MACRO:
12  *
13         TEST
14  *
15         TEST,0
16  *
17         TEST            9
18         END
```

Figure 3-5. Listing of ARGMIS with LALL in Effect

```
POS ARGPAR
LIST

 1   * MACRO ARGUMENT PARAFORMS
 2   *
 3   * DEFINE TWO MACROS
 4   * NAMED CALL AND LAST:
 5   *
 6   CALL  MAC
 7         BLM,%1          %2
 8         DFC             %3
 9         DFC             %4
10         EMP
11   *
12   LAST  MAC
13         BLM,%1          %2
14   A     SET             %0
15   B     SET             %A
16         EMP
17   *
18         EXT             SUB1
19   *
20   * SOME CALLS TO THESE MACROS:
21   *
22         CALL,7,SUB1,10,20
23   *
24         LAST,1,2,3
25   *
26         LAST,#1,#2,#3,#4,#5,#6,#7,#8,#9
27   *
28         END
```

Figure 3-6. Listing of ARGPAR with LALL In Effect

```
POS ARGPAR
LNT
LIST

1------------
    28         END
```

Figure 3-7. LIST Command with LNT in Effect

```
LNT
LAE ARG+++ LIS

   1----ARGPAR
        28         END
TOTAL RECORDS WRITTEN =     29/       3
   2----ARGMIS
        18         END
```

Figure 3-8. LAE Command with LNT in Effect

LST

Modify Output Listings

The LST command is similar to LNS, LNT, and LNAME in that it is used prior to any command that generates listings of source files to the LO file (e.g., LIST and LAE with the LIST parameter). With LST, you must define one or more character strings. Then when you use any command that generates listings of source files to the LO file, the listings will contain only those source lines that have the user-specified strings in them.

To override the effect of a previous LST, use the LALL command. LALL is the default mode when SED is first invoked.

SYNTAX

[.] LST (string1) [(string2)] ....

Each character string must be enclosed in any of the valid string delimiters that were listed in Section 3.1.

Figure 3-9 is a listing of a source file named ARGMIS that was generated by a LIST command with LALL in effect (the default mode when SED is first invoked). In Figure 3-10, an LST command was used with the string "TEST" as a parameter. Now note that when another LIST command is used, the source listing only contains those lines that have the string "TEST" in them.

```
POS ARGMIS
LALL
LIST

 1  * ASSEMBLER CONDITIONAL DIRECTIVES
 2  *
 3  TEST  MAC
 4        IFP           %1,A
 5        DFC           5
 6  A     IFM           %1,DONE
 7        DFC           %1
 8  DONE  AOP
 9        EMP
10  *
11  * SOME CALLS TO THE MACRO:
12  *
13        TEST
14  *
15        TEST,Ø
16  *
17        TEST          9
18        END
```

Figure 3-9. LIST Command with LALL in Effect

```
POS ARGMIS
LST /TEST/
LIST

1------------
     3  TEST  MAC
    13        TEST
    15        TEST,Ø
    17        TEST          9
```

Figure 3-10. LIST Command with LST in Effect

MACRO

Define, List, or Remove Macros

The MACRO command defines the start of a macro definition, lists one or more current macro definitions, or removes a macro.

A macro definition consists of the following components:

1. A MACRO command followed by the user-specified name.
2. The sequence of SED commands that will be executed when the macro is invoked.
3. An ENDMACRO command.

When you are defining a macro and SED encounters a MACRO command, it reads subsequent lines input from the CI file and puts them into the macro stack until an ENDMACRO command is encountered. To invoke the macro, simply enter the macro name just as you would a SED command.

The macro name can be the name of an existing SED command if you want to redefine a standard SED command. When the macro is invoked, SED first checks a list of macro names before checking its standard list of commands.

Macro definitions can contain calls to other macros up to a nesting level of four, but a macro cannot call itself. To cause a macro to execute continually until SED reaches the beginning-of-program (BOP) or end-of-program (EOP), place a CONTINUE command on the line immediately before the ENDMACRO command (see the example under the CONTINUE command).

To list the definitions of all current macros, use the MACRO command without any parameters. Listings of macro definitions are written to the CO file.

SYNTAX

To define a macro:

[.] MAC[RO] name

To list a macro definition:

[.] MAC[RO] name LIS[T]

To remove a macro definition:

[.] MAC[RO] name REM[OVE]

To list all macro definitions:

[.] MAC[RO]

name Parameter 1 is the name of the macro you want to define, list, or remove. Macro names can be from one to three CAN-code characters in length. A name must be specified to define a macro or to use either of the keywords described for Parameter 2. If MACRO is used without any parameters, SED lists the definitions of all current macros to the CO file.

Parameter 2 can only be used if a name was specified in Parameter 1. Valid keywords for Parameter 2 are:

LIS[T] Lists the definition of the macro specified in Parameter 1.

REM[OVE] Removes the definition of the macro specified in Parameter 1.

EXAMPLES

```
MAC LAI
CHA :LDI:LAI:
END
```

This macro changes the string LDI to LAI on the current line. To locate a source line that contains the string LDI and then invoke the macro, you would enter:

```
G 'LDI'
LAI
```

MACRO LAI LIST

Lists the definition of the LAI macro to the CO file.

MACRO

Lists the definitions of all current macros to the CO file.

MAC LAI REM

Removes the definition of the macro named LAI that was just defined above.

MARK

Define EOF Marks

The MARK command is used to specify file marks to separate or subdivide source programs into subprograms. The defined mark(s) will be processed by most SED commands in the same way as the default file mark ($$). One exception is the TRANSFER command, which ignores any user-defined file marks and always uses the physical end-of-file ($$) as it file mark.

If neither mark is defined as $$ or END, SED assumes that any source line with "string1" starting in column one is the first line of a program, and any line starting with "string2" in column one is the first line of a subprogram.

There are special cases if either string is $$ or END. If either string is $$, SED interprets it as the system end-of-file mark (i.e., the end of a program). If either string is END, SED considers the mark to be a FORTRAN END statement. Any END statement within a program will then be treated as the last source line of a program or subprogram.

If "string1" is $$, then "string2" (if specified) must be $$ or END. If "string1" is END, then "string2" (if specified) must be $$.

The default for the MARK command is:

```
MARK '$$' '$$'
```

which assumes that each source file is one program and that subprograms do not exist.

If you use the MARK command to define your own file marks, it must be used while you are in sequential mode.

Refer to Section 2.8 in Chapter 2 for more information on user-defined file marks. To list or copy subprograms after the marks have been defined, refer to the LIST and COPY command descriptions in this chapter. You can also position to a specific subprogram within a program using the POSITION command after you define subprogram marks. Refer to the description of the POSITION command in this chapter for more information.

SYNTAX

```
[.] MAR[K] (string1) [(string2)]
```

Each string must be enclosed in any of the valid string delimiters that were listed in Section 3.1.

Modify Characters within a Source Line

The MODIFY command is used to insert, delete, or overlay characters within a source line. If the NOQUICK command or option is in effect, SED will display the line on the CO file after it is modified.

If you are editing in sequential mode, SED copies the original source records from the current position of the SI file to the current position of the SO file, up to and including the source line that is being modified. In both random and sequential modes, the newly modified line becomes the new current line.

With a MODIFY, SED displays the source line indicated by the SLR parameter on the CO file. You can then overlay, delete, or insert characters by entering the new information at the appropriate column positions below the displayed line.

Any characters that are skipped over (with the space bar) remain intact.

To overlay characters, simply enter the new characters under the corresponding characters of the displayed line.

To delete characters, enter an underscore (_) under the corresponding characters of the displayed line. If you want an underscore to appear as part of the text line, you have to redefine the underscore character using the DEFINE UNDERSCORE command.

To insert characters, enter an uparrow (^) below the character where the new text is to be inserted. The new characters will be inserted before the character under which the uparrow appears. After entering the inserted text, enter another uparrow, an underscore, or a blank space to terminate the insert.

To insert an actual blank space into the source line or to have the inserted text contain a blank space, enter an ampersand (&) to represent the blank space character.

If the line that is displayed is not the line you want to modify, press the carriage return and the line will remain as it was before the MODIFY. However, if a REVISION command is in effect, the revision string will still be inserted into the line. Refer to the description of the REVISION command in this chapter for more information.

Other commands that can be used to modify source lines are OVERLAY, CHANGE, IST, and MST. For more information, refer to Section 2.5.6 in Chapter 2 or the descriptions of those commands in this chapter.

SYNTAX

[.] MODIFY [SLR]
MOD
M

[SLR] (OPTIONAL) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. If not specified, the default is to display the current line so that it can be modified.

EXAMPLES

.MOD 10

Displays line 10 so that it can be modified.

M

Displays the current line so that it can be modified.

MOD 6"MAC"

Displays the first source line encountered that has the character string MAC starting in column six. After the line is displayed, it can be modified.

Move Source Lines

The MOVE command can only be used in random mode. It is used to move one or more lines from one location within a source file to another location within the same file. The line(s) to be moved are inserted after the current line and then deleted from their original location.

The DUPLICATE command can be used to reproduce one or more lines and then place them after the current line without deleting them from their current location.

SYNTAX

[.] MOV[E] SLR1 [SLR2]

SLR1 (REQUIRED) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. It specifies the first line in a range or the only line to be moved.

[SLR2] (OPTIONAL) Parameter 2 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. SLR2 is optional only if you want to move the single line indicated by SLR1. To move a range of lines, SLR2 is required and should specify the last line in the range of lines to be moved.

EXAMPLES

MOV 'SUBR1','SUBR1'+8

Moves the range of lines from the first line encountered that contains the string SUBR1 through the next eight lines, and places them after the current line.

.MOVE 1<ABC>,1<ABC>+9

Moves the range of lines starting at the first line encountered that contains the character string ABC in column one through the next nine lines, and places them after the current line.

MOVE 8

Moves line eight and places it after the current line.

Delete Character String from Current Line

The MST command removes a specified character string from the current line. If a string is not specified, SED uses the character string from the most recent MST command. If an MST command has not been used or if a DST command has been used since the last MST, SED uses the string from the most recent DST command.

If you have already defined a string with the DST command and then use MST with a string, that string then becomes the currently defined string, just as if you had defined it with the DST command. Subsequent use of an IST command (to insert a string) will then use this newly defined string. The current string, whether defined in a DST or MST command, is remembered until you exit SED or until it is redefined by another DST or MST command.

If the NOQUICK command or option is in effect, SED displays the modified line after the MST command is processed.

For more information and examples, refer to Section 2.5.6.4 in Chapter 2 or to the descriptions of the DST and IST commands in this chapter.

SYNTAX

[.] MST [(string)]

(string) Parameter 1 is optional if you want to delete the currently defined string from the current line. If you specify a string, that string is removed from the current line and also becomes the currently defined string. The string must be enclosed in any of the valid string delimiters that were listed in Section 3.1.

EXAMPLES

MST "ABC"

Removes the string ABC from the current line. That string now becomes the currently defined string for subsequent IST commands and MST commands used without a string.

DST /ABC/
MST

Removes the string ABC from the current line. The currently defined string for subsequent IST commands and MST commands specified without a string remains ABC.

DST /ABC/
...enter other SED commands...
MST /END/

Removes the string END from the current line. The currently defined string for subsequent IST commands and MST commands specified without a string is now END, not ABC.

Copy Source Lines to SO File

The NGE and IGE commands can only be used in sequential mode. They are used to copy selected source lines from the SI file to the current position of the SO file. After the lines are copied, SED writes an end-of-file mark on the SO file unless the NOFILE command is in effect.

The only difference between the use of NGE and IGE is that IGE performs a BKFILE operation on the SO file prior to copying the specified lines.

SYNTAX

[.] NGE SLR1 [SLR2]

and

[.] IGE SLR1 [SLR2]

SLR1 (REQUIRED) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. It specifies the first line or only line to be copied to the SO file.

[SLR2] (OPTIONAL) Parameter 2 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. SLR2 is optional if you only want to copy the single line indicated by SLR1. To copy a range of lines, SLR2 is required and should specify the last line in the range of lines to be copied to the SO file.

EXAMPLES

NGE 7,200
Copies lines seven through 200 to the SO file.

.IGE C,C+95
Performs a BKFILE operation on the SO file and then copies the range of lines from the current line through the next 95 lines to the SO file.

IGE 'XYZ','XYZ'+400
Performs a BKFILE operation on the SO file and then copies the range of lines from the first line encountered that contains the string XYZ through the next 400 lines to the SO file.

NOCOMPRESS

Write to SO File in Uncompressed Format

The NOCOMPRESS command causes all subsequent records that are written to the SO file to be in uncompressed format. Uncompressed records contain up to 80 ASCII bytes per record.

The NOCOMPRESS command itself does not actually write any records to the SO file, but has an effect on other SED commands that do write to the SO file.

The opposite of the NOCOMPRESS command is COMPRESS, which is the default mode when SED is first invoked. The COMPRESS command is normally used to return to writing compressed records to the SO file after a NOCOMPRESS was previously used.

Note that if records are copied to the SO file with the TRANSFER command, the NOCOMPRESS command, if in effect, is ignored. TRANSFER always copies records in compressed format, although the records being copied can be compressed or uncompressed. To copy records to the SO file in uncompressed format, use the NOCOMPRESS and COPY commands as shown in the example below.

SYNTAX

[.] NOC[OMPRESS]

EXAMPLES

```
NOCOMPRESS
COPY
```

In random mode this example would copy all of the records from the WRK file to the SO file in uncompressed format. In sequential mode this example would copy the program from the current position of the SI file to the SO file in uncompressed format.

NOFILE

Suppress Writing of EOF Marks on SO File

If the NOFILE command is in effect when source records are copied to the SO file, end-of-file (EOF) marks will not be written to the SO file when they are encountered on the SI file.

To return to writing file marks on the SO file after NOFILE has been used, use the FILE command. FILE is the default mode when SED is first invoked.

SYNTAX

[.] NOF[ILE]

NOPACK

Stop Effect of PACK Command

The NOPACK command stops the effect of the PACK command. PACK removes comments from the statements in Assembly language source programs. NOPACK is the default mode when SED is first invoked.

Refer to the description of the PACK command for more information on the rules SED uses to remove Assembly language comments from source programs.

SYNTAX

[.] NOP[ACK]

NOQUICK

Automatic Display of Source Lines

The NOQUICK command causes SED to automatically display source lines on the CO file after an OVERLAY, CHANGE, MODIFY, IST, MST, SKIPTO, GOTO, FORWARD, or BACKWARD command, and also after you position to a line by entering only a line number. The NOQUICK command performs the same function as the NOQUICK option.

The QUICK command or option suppresses the automatic display of source lines affected by these commands. QUICK is the default mode when you first invoke SED.

SYNTAX

[.] NOQ[UICK]

or

[.] OPT[ION] NOQ[UICK]

Suppress Generation of Sequence Numbers

The NOSEQUENCE command suppresses the generation of sequence numbers that was started by a SEQUENCE or EXSEQUENCE command. It will also stop the effect of a previous UNSEQUENCE command, which places blanks in columns 73 through 80 of each source line. NOSEQUENCE is the default mode when SED is first invoked.

If source lines have already been copied to the SO file, NOSEQUENCE will not remove the sequence numbers already in place. Also, if a program has already been cataloged with sequence numbers, NOSEQUENCE will not remove them. In sequential mode NOSEQUENCE will only prevent sequence numbers from being placed on subsequent records that are copied to the SO file. In random mode NOSEQUENCE will remove all sequence numbers from the source lines on the WRK file if specified before you catalog or recatalog the program.

For more information on the use of sequence numbers, refer to Section 2.12 in Chapter 2 or to the descriptions of the SEQUENCE, EXSEQUENCE, and UNSEQUENCE commands in this chapter.

SYNTAX

[.] NOS[EQUENCE]

Insert New Records into a Source File

The NPU and IPU commands are used to insert new records into a source file. Normally, SED reads the text input for additions from logical file CI or AI or an insert file. If NPU or IPU is used, the text input for the addition must be located on the logical file that was named in a previous PRF command.

The only difference between NPU and IPU is that IPU performs two BKFILEs followed by an AVFILE on the PRF file before reading the records on that file.

SED reads the text input from the PRF file until it encounters an end-of-file mark ($$) or a record that contains a terminator character (the same character that is used to terminate input for the ADD and REPLACE commands). By default, the terminator character is a period starting in column one. You can also redefine the terminator character with the DEFINE command (using the TERMINATOR parameter).

If NPU or IPU is used in sequential mode, SED copies the original source records from the current position of the SI file to the current position of the SO file, up through the source line after which the new records are to be inserted. Records are then read from the PRF file until SED encounters an EOF mark or a terminator character as described above.

In random mode SED inserts the source lines from the PRF file directly into the WRK file after the specified line. Again, an EOF mark or terminator character is needed to terminate the addition.

SYNTAX

[.] NPU SLR

and

[.] IPU SLR

SLR (REQUIRED) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3 that indicates the line after which new records are to be added. To add new records before line one, use a zero for the SLR parameter (e.g., NPU 0 or IPU 0).

EXAMPLES

.IPU 20
Performs two BKFILEs and one AVFILE on the PRF file and then inserts the additions from the PRF file after line 20.

NPU 'RTN'+3
Inserts the additions from the PRF file three lines after the line that contains the string RTN.

NPU C+7
Inserts the additions from the PRF file seven lines after the current line.

Replace Records in a Source File

The NRE and IRE commands are used to replace lines in a source file with text input from the logical file named in a previous PRF command. The only difference between NRE and IRE is that IRE performs two BKFILEs followed by an AVFILE on the PRF file before reading the records from that file.

SED reads the text input from the logical file named in the most recent PRF command until it encounters an end-of-file mark ($$) or a record that contains a terminator character (the same character used to terminate input for the ADD and REPLACE commands). By default, the terminator character is a period starting in column one. You can also redefine the terminator character with the DEFINE command (using the TERMINATOR parameter).

If NRE or IRE is used in sequential mode, SED copies the original source records from the current position of the SI file to the current position of the SO file, up to the source line to be replaced. This line, and any others if a range of lines is to be replaced, are then deleted. SED then reads the replacement records from the PRF file until an EOF mark or a terminator character is encountered as described above.

In random mode SED deletes the specified line or range of lines from the WRK file and then inserts the replacement lines from the PRF file into the WRK file. Again, an EOF mark or terminator character is needed to terminate the addition.

SYNTAX

[.] NRE SLR1 [SLR2]

and

[.] IRE SLR1 [SLR2]

where SLR1 and SLR2 are source line references as previously described in Sections 2.4.2.1 through 2.4.2.3. If SLR2 is not specified, only the single line indicated by SLR1 is replaced. To replace a range of lines, specify the first line as SLR1 and the last line in the range as SLR2.

EXAMPLES

NRE 50,200
Replaces lines 50 through 200 with the text input from the PRF file.

.IRE C,C+200
Performs two BKFILEs and an AVFILE on the PRF file and then replaces the range of lines from the current line through the next 200 lines with the text input from the PRF file.

NRE :SUB1:,:SUB1:+70
Replaces the range of lines from the first line encountered that contains the string SUB1 through the next 70 lines with the text input from the PRF file.

OPTION

Set or Reset SED Options

The OPTION command sets or resets SED options that control various characteristics of the SED operating environment.

SYNTAX

[.] OPT[ION] option [option] ...

where option is one or more of the following keywords:

PRO[MPT]/NOP[ROMPT]	Displays or inhibits the display of a prompt character to indicate that SED is ready to accept input. The default prompt character is a period. **Default = NOPROMPT**
HOL[D]/NOH[OLD]	Controls whether or not SED enters a hold state after an error occurs. **Default = HOLD**
ONE/TWO	ONE suppresses the display of line numbers for source lines that are written to the CO file. TWO displays line numbers. For the DISPLAY command, a screenful of text is 20 lines if ONE is set and 10 lines if TWO is set. TWO is used with the NST option to affect the output of the PRINT command. This combination causes the line number and text of the line to be displayed on a single line. If ONE is set or if TWO and STD are set, PRINT displays the line number on one line and the text of the source line on the next line. **Default = TWO**
LIM[IT]/NOL[IMIT]	LIMIT must be set if the LIMIT command is used, which prevents certain columns of each source line from being modified. NOLIMIT allows any characters in every source line to be modified. The LIMIT command has no effect if the NOLIMIT option is set. **Default = NOLIMIT**
DAT[E]/PGM	Saves the date and time or the program name as part of the information about a source program when it is cataloged in a USL. The PGM option uses the name as found on the first PGM or PROGRAM statement within the source file. Either the date and time or the program name is displayed under the IDENTIFICATION column of a directory listing. If the PGM option is set and there is no PGM or PROGRAM statement in the file, this field will be left blank in a directory listing. **Default = PGM**
TAB/NOT[AB]	Process tabs (TAB) or treat them as normal ASCII characters (NOTAB). **Default = TAB**
SCR[EEN]/NOS[CREEN]	Displays output from the LIST comand a screenful at a time (SCREEN) or generates uninterrupted output to the terminal (NOSCREEN). **Default = NOSCREEN**

MET/MCO	MET terminates processing if errors occur during macro execution. MCO continues processing if errors occur during macro execution. **Default = MET**
QUI[CK]/NOQ[UICK]	Displays (QUICK) or inhibits the display (NOQUICK) of output lines modified by an OVERLAY, CHANGE, MODIFY, IST, or MST command or accessed by a SKIPTO, GOTO, FORWARD, BACKWARD, or line number command. The QUICK and NOQUICK commands perform the same functions as the QUICK and NOQUICK options. **Default = QUICK**
STD/NST	NST is used in conjuntion with option TWO. If the NST and TWO options are in effect and the record size of the CO file is 82 bytes or longer, source lines listed to the CO file by a PRINT command appear on one line instead of two. If STD is set, PRINT output to the CO file is in standard ASCII format (i.e., the line number and text of the record appear on two lines). Refer to the PRINT command for samples of these two formats. **Default = STD**

EXAMPLES

```
OPTION PROMPT,NOH,NOQ

OPT LIM NST PRO DATE
```

OVERLAY

Overlay Characters within a Source Line

OVERLAY is one of the commands that can be used to modify selected characters within a source line. Other commands that can be used for this function are CHANGE, MODIFY, IST, and MST. OVERLAY is generally used to make changes based on character position. If characters are to be added or deleted from a line, it is probably easier to use CHANGE, MODIFY, IST, or MST. Refer to Section 2.5.6 in Chapter 2 for a more complete discussion on the various ways to modify characters within a source line.

SYNTAX

[.] OVERLAY [(string)] [integer]
OVE
O

[(string)] (OPTIONAL) Parameter 1 consists of the replacement characters in their respective column positions. For example, if you only want to replace characters in the second and third columns of a source line, then the first character of the string must be a blank space so that column one will not be overwritten. If a character string is not specified, SED uses the string from the most recent OVERLAY command.

[integer] (OPTIONAL) Parameter 2 specifies the number of lines on which the overlay is to take place, starting on the current line. If not specified, the overlay takes place only on the current line.

As an example, suppose that the current line of the source file you are editing contains a misspelled word:

```
C TIHS IS A COMMENT.
```

To correct the string "TIHS" to "THIS" using OVERLAY, you would enter the command as follows:

```
OVERLAY (␣␣THIS)
```

where ␣ represents a blank space. Note that because the character "T" in the string "TIHS" starts in column three, two blank spaces are required to skip over columns one and two.

To overlay an actual blank space, enter an ampersand (&) in the column position where it is to appear. For example:

```
OVERLAY (&&ABC)
```

would overlay columns one and two of the current line with blank spaces and columns three, four, and five with the string ABC.

If you want the & to appear as an actual character, you must first redefine the space character with the DEFINE command (using the SPACE parameter).

The tab character can be used when specifying the character string for the OVERLAY command. This is useful for making changes based on column position as shown in the examples below.

EXAMPLES

The following example will overlay the first three characters after the third tab position on the current line with blank spaces:

```
DEF TAB :^:
O :^^^&&&:
```

Given that the string SMITH appears beginning in column 20 of a source line and the first tab stop is set at column 20, the following sequence of commands would be used to change the string SMITH to JOHNSON on the current line:

```
DEF TAB :\:
O "\JOHNSON"
```

PACK

Remove Comments from Assembler Programs

The PACK command removes comments from statements in Assembly language source programs. This feature is useful for compressing macro files to conserve file space.

When PACK is in effect, source statements are modified or deleted according to the following rules:

1. Any statement that begins with an asterisk (*) in column one will be deleted.

2. TTL and PGM statements will not be modified.

3. Any statement that contains a semicolon (;) or that begins with a percent sign (%) in column one will not be deleted or modified.

4. The presence of 15 or more blank spaces on a statement will terminate that source record unless those spaces are part of a character string that is enclosed in single or double quotes.

5. Any fields after the first three (label, opcode, and argument) will be deleted. The label field begins in column one and is terminated by a blank. The opcode field begins after the label field or in the first nonblank column after column one if the label field is blank. The opcode field may contain one or more arguments separated by commas. The opcode field is terminated by one or more blanks that are not part of a CAN-code argument or character string. The argument field begins after the opcode field and is treated in the same way. Anything after the argument field is deleted if PACK (without the 15 option) is in effect.

The effect of the PACK command can be stopped by using the NOPACK command. NOPACK is the default mode when SED is first invoked.

SYNTAX

[.]PAC[K][15]

[15] (OPTIONAL) Parameter 1 modifies the effect of PACK so that source lines will only be terminated for reasons 1 and 4 as listed above.

EXAMPLES

The following examples are sample source statements from an Assembly language program. Assuming that the PACK command has been set, these examples are designed to show the effect that PACK would have.

*ABCDE

This statement will be deleted because it begins with an asterisk in column one.

%ABCDE

This statement will not be modified because it begins with a percent sign in column one.

LABEL TBMB,R3,R5

This statement will not be modified at all because it does not contain an asterisk in column one and the statement contains three valid fields (label, opcode, and arguments).

LABEL IF @%3=@A,+2 XYZ

XYZ will be deleted because it appears after the argument field.

LABEL IF @%3 = @ A,+2 XYZ

= @ A,+2 XYZ will be deleted because a blank space appears after the third field (@%3). This will generate an error. The source line should be changed if the PACK command is to be used.

PAUSE

Send a Message to CO File and Hold

PAUSE is one of three commands used to send a message to the CO file for the operator's attention. The other two are ACTION and NOTE. PAUSE and ACTION both cause SED to enter a hold state after the message. Execution can be resumed through the Operator Communication (OC) task. With NOTE, SED does not enter a hold state after the message is displayed, but continues to read the next command from the CI or AI file.

The use of ACTION or NOTE over PAUSE is recommended because of their Job Control equivalents ($ACTION and $NOTE).

SYNTAX

[.] PAU[SE] message

[.] ACT[ION] message

[.] NOT[E] message

EXAMPLES

PAU ATTENTION: MOUNT MAGTAPE 1

ACTION ATTENTION: MOUNT MAGTAPE 1

NOTE END OF JOB STREAM

It is recommended to spell out the command completely to avoid confusion. For example, if NOTE was abbreviated, the message on the CO file would read:

NOT END OF JOB STREAM

Print Source Lines with Control Characters

The PC command is used to list statements from a source program. It is the same as PRINT except that PC displays a C above any column of text where a control character is located. The display of statements is written to logical file CO.

SYNTAX

[.] PC [integer]

[integer] (OPTIONAL) Parameter1 specifies the number of source lines to print that contain control characters, starting at the current line. If not specified, the default is the current line.

All of the information about PRINT command in Section 2.6.1 and in the command description in this chapter is also applicable to the PC command. Refer to either section for more information.

POSITION

Locate Source Programs on the SI File

The POSITION command is used to locate a particular source program on the SI file so that you can edit, copy, list, or otherwise manipulate it. After positioning to the appropriate file, you can then use the RANDOM command to copy the file to the WRK file so that you can edit it randomly.

If the SI file has a directory and you have used the default file mark ($$) to separate source programs from one another, SED locates the program by its cataloged name in the USL directory. The pointer on the SI file is then set to the first record of the program. When you enter random mode, SED will then copy the source file to the WRK file.

If the SI file does not have a directory and the file mark used to separate source programs is either $$ or a FORTRAN END statement, SED searches the SI file sequentially. The name1 parameter must then specify the name of the program as found on a PGM, PROGRAM, BLOCKDATA, SUBROUTINE, FUNCTION, or OVERLAY statement within the program. When SED locates the identifying source line, it backspaces to the beginning of that program.

By default, the SI file is rewound before SED searches for the specified program. The NOREWIND parameter can be used if the SI file is assigned to a magnetic tape. This will prevent the tape from being rewound.

If the MARK command has been used to define a file mark other than $$ or END, SED will interpret any source line with that mark in column one as the first record of a new program. If the SI file does not contain a directory and you want to position to a program that uses alternative file marks, you must place the program name on the same line following the characters of the mark. This name serves to identify the program in the same way as a PGM, PROGRAM, BLOCKDATA, SUBROUTINE, FUNCTION, or OVERLAY statment as described above. For more information about alternative file marks, refer to Section 2.8 in Chapter 2 or the description of the MARK command in this chapter.

SYNTAX

[.] POS[ITION] name1 [name2] [integer] [NOREWIND]

name1 (REQUIRED) Parameter 1 specifies the name of the program you want to locate on the SI file.

[name2] (OPTIONAL) Parameter 2 specifies the name of a subprogram within the program name1 that you want to locate on the SI file. This feature can only be used if you have subdivided programs into subprograms by defining a second file mark with the MARK command.

[integer] (OPTIONAL) Parameter 3 should only be used if the SI file does not contain a directory. If used, it specifies that SED should search for the named program or subprogram sequentially until a specific number of file marks have been bypassed.

NOR[EWIND]

(OPTIONAL) Parameter 4 prevents the SI file from being rewound if it is assigned to a magnetic tape.

EXAMPLES

POS PAYROLL

Positions the SI file to the source program named PAYROLL.

.POS PROG1,5

Searches the SI file sequentially for the source program named PROG1. The search continues until SED bypasses five file marks. If the program is not found after bypassing five file marks, SED will return an error message.

.POSITION PROG,SUBPROG

Positions the SI file to the subprogram named SUBPROG within the source program named PROG.

PRF

Name the File for Subsequent NPU/IPU and NRE/IRE Commands

The PRF command is used to name the logical file from which text input will be read when source lines are added with the NPU or IPU command or replaced with the NRE or IRE command.

If an NPU, IPU, NRE, or IRE command is used without defining the input file through the PRF command, the assignment defaults to the AI file.

SYNTAX

[.] PRF file

file (REQUIRED) Parameter 1 is the name of the logical file from which the input text for subsequent additions (NPU or IPU) and replacements (NRE and IRE) should be read.

Display Source Lines

The PRINT command displays lines from a source file, beginning with the current line, on the CO file. The current line pointer is not affected by this command.

The characters "BOP" are displayed if you try to print source lines that precede the first line of the file. The characters "EOP" are displayed if you try to print source lines that are beyond the last line of the file.

PRINT displays records in one of two format, depending on the record size of the CO file:

```
aaaa zzzzzzzz
bbb ----------------------------------------> bbb

aaaa
bbb ----------------------------------------> bbbzzzzzzzz
```

where aaaa is the line number; bbb --> bbb are the first 72 characters of the source line; and zzzzzzzz are the last eight characters of the source line.

The first line of either format can be suppressed by setting option ONE:

```
OPTION ONE
```

The default option is TWO, which displays two lines of output for every source line that is printed.

If the NST and TWO options are in effect and the record size of the CO file is 82 bytes or longer, then the line number and text of the source line are printed on one line in the format:

```
aaaa bbb ----------------------------------> bbbbzzzzzzzz
```

where aaaa is the line number; bbb --> bbb are the first 72 characters of the source line; and zzzzzzzz are the last eight characters of the source line.

If the source file contains control characters, use the PC command instead of PRINT. PC displays source lines in the same format as PRINT but also indicates where control characters are located.

SYNTAX

```
[.] PRINT [integer]
    PRI
    P
```

[integer] (OPTIONAL) Parameter 1 is the number of lines to print, beginning at the current line. If not specified, SED prints the current line.

PTA
QUICK

PTA

Display Current Tab Settings

The PTA command reports the column positions where tabs are currently set. You can also use the SCALE command to print a scale of numbers from one through 79 with a T above each column where tabs are currently set.

To clear tabs, use the CLEARTAB command. To set tabs, use the SETTAB command.

SYNTAX

[.] PTA

QUICK

Suppress Automatic Display of Source Lines

The QUICK command suppresses the automatic display of source lines to the CO file after an OVERLAY, CHANGE, MODIFY, IST, MST, SKIPTO, GOTO, FORWARD, or BACKWARD command. It also suppresses the printing of the the current line when you position to it by entering only a line number. The QUICK command performs the same function as the QUICK option and is the default mode when you first invoke SED.

To request an automatic display of source lines, use the NOQUICK command or set the NOQUICK option.

SYNTAX

[.] QUI[CK]

or

[.] OPT[ION] QUI[CK]

Remove Entries from a Directory

The RAE command removes specific programs from the USL directory. Wildcard characters can be used when specifying the program to be removed, which enables you to remove more than one program at a time. The REMOVE command can also be used to remove entries from the directory of a USL file, but wildcard characters cannot be used with REMOVE.

Wildcard characters are specified in the form of plus signs. A wildcard matches any character in that position.

SYNTAX

[.] RAE name

name (REQUIRED) Parameter 1 is the name of the program to be removed from the USL directory, using wildcard characters (plus signs) to represent any character in that position.

EXAMPLES

RAE F+++++++
Removes all programs whose names are eight characters in length and that begin with the character F.

.RAE TEST+++
Removes all programs whose names are eight characters in length and that begin with the four characters TEST.

RAE +++OLD
Removes all programs whose names are six characters in length and that end with the three characters OLD.

RANDOM

Enter, Exit, or Recover Random Mode

The RANDOM command is used to enter or exit random mode or to recover the most recent program that was on the WRK file. The RECOVERY option is useful if your SED session was inadvertently interrupted while you were editing in random mode. The EXIT option is useful if you want to exit random mode and still remain in SED (e.g., to examine other source files). Specifying RANDOM EXIT has the same effect as using the XRN command.

To enter random mode, use the RANDOM command without any parameters.

Commands such as CATALOG, RECATALOG, and COPY will return SED to sequential mode from random mode after they are processed.

SYNTAX

[.] RAN[DOM] [keyword]

To enter random mode, do not specify any parameters with the RANDOM command. To exit random mode or recover the most recent WRK file, use one of the following keywords for Parameter 1:

EXI[T] To exit random mode and return to sequential mode.

REC[OVERY] To recover the most recent source program that was on the WRK file.

EXAMPLES

RAN REC
Recovers the most recent source file from the WRK file.

RANDOM
Enters random mode and copies the source program from the current location of the SI file to the WRK file.

RAN EXI
Exits random mode.

Recatalog Programs in a USL

The RECATALOG command is used to catalog the most recent version of an existing source file in a USL. This command is used if you have made changes to an existing file and want to replace the previous version with the newly modified version. In sequential mode SED expects to find the source file on the SI file. In random mode the source file is copied from the WRK file. If you recatalog the file while in random mode, SED returns to sequential mode after the command is processed.

To recatalog a file, SED copies it into the largest available space in the USL and enters it into the directory under the name #FAFFFAFFFAFF. Once it is cataloged under this name, SED deletes the source file that was named on the RECATALOG command and renames the source file #FAFFFAFFFAFF to whatever name was specified in the RECATALOG command.

If you used the POSITION command to initially locate the source file on the SI file, you can use the * parameter instead of specifying the source file name. SED then recatalogs the file with the name used in the most recent POSITION command.

SYNTAX

```
[.] RECATALOG
    RECAT      { name }   [id]
    REC        { *    }
```

Parameter 1 is required and can be one of the following:

name The name that was used when the file was first cataloged.

* To recatalog the file with the name used in the most recent POSITION command.

[id] (OPTIONAL) Parameter 2 can be used if Parameter 1 was either a name or an asterisk. It is a string of one to eight characters that can be used as an identification for the source program. The identification will be cataloged with the source program and the PGM or DATE option (whichever is set) has no effect. The identification string will be displayed under the IDENTIFICATION column in a directory listing (see Figure 3-2 under the LDE command).

EXAMPLES

RECATALOG MYFILE
Recatalogs the source file named TEST.

.REC MAXIV,NEWREV
Recatalogs the source file named MAXIV and assigns the identification NEWREV to it.

REC *
Recatalogs the source file with the name that was specified in the most recent POSITION command.

REDEFINE

Rename an Attached File

The REDEFINE command renames a previously attached file.

SYNTAX

[.] RED[EFINE] name1 name2

name1 (REQUIRED) Parameter 1 is the name of the currently attached file.

name2 (REQUIRED) Parameter 2 is the new name to be assigned to the file specified in Parameter 1.

EXAMPLES

REDEFINE SC3 SC4
Renames the attached file SC3 to SC4.

.RED AT1,AT2
Renames the attached file AT1 to AT2.

Remove an Entry from a Directory

The REMOVE command removes the entry for a source file from the USL directory, in effect deleting that source file.

To remove more than one directory entry at a time using wildcard characters, use the RAE command.

SYNTAX

[.] REM[OVE] name

name (REQUIRED) Parameter 1 is the name of the cataloged source program to be removed from the USL directory.

EXAMPLES

.REMOVE AB45
Removes the directory entry for the source file named AB45.

REM XYZ
Removes the directory entry for the source file named XYZ.

RENAME

Rename a Source File

The RENAME command is used to rename a source file. When this command is used, SED changes the entry for the named file in the USL directory. The source file can no longer be referenced by its previous name.

SYNTAX

[.] REN[AME] name1 name2

name1 (REQUIRED) Parameter 1 is the current name of the cataloged source file.

name2 (REQUIRED) Parameter 2 is the new name to be given to the source file specified in Parameter 1.

EXAMPLES

RENAME ALPHA,ALPHA2
Changes the name of the source file ALPHA to ALPHA2.

.REN BEE,CEE
Changes the name of the source file BEE to CEE.

Replace Source Lines

The REPLACE command is used to replace one or more source lines within a file with one or more new lines of text. By default, SED reads the text input for the replacement records from the CI file. If the AO options is set, then SED reads the replacement records from the AI file. If an INSERT command was used, then SED reads the replacement records from the logical file named on that command.

When REPLACE is used in sequential mode, SED copies the original source records from the current position of the SI file to the current position of the SO file until the source statement specified by the SLR1 parameter is reached. SED deletes this line, and the other lines if a range of lines is to be replaced. SED then reads text input from the CI, AI, or insert file to replace those lines that were deleted.

In random mode SED deletes the specified line or range of lines directly from the WRK file and inserts the additions from the AI, CI, or insert file.

A REPLACE must be terminated by an end-of-file ($$) mark or a terminator character. By default, the terminator character is a period starting in column one. This same character is also used to terminate the addition of new records by an ADD command. You can redefine the terminator character with the DEFINE command (using the TERMINATOR parameter).

For more detailed information on using the REPLACE command, refer to Section 2.5.4 in Chapter 2.

The NRE or IRE commands can also be used to replace source lines with text from the file named in a previous PRF command. Refer to the descriptions of the NRE/IRE and PRF commands in this chapter for more information.

SYNTAX

```
[.] REPLACE [SLR1 [SLR2]]
    REP
    R
```

where SLR1 and SLR2 are source line references as previously described in Sections 2.4.2.1 through 2.4.2.3. If both parameters are specified, the range of lines from SLR1 through SLR2 are replaced. If neither parameter is specified, only the current line is replaced.

EXAMPLES

REP C+7
: Replaces the line that is seven lines after the current line.

REPLACE C,'END'
: Replaces the range of lines from the current line through the first line encountered that contains the string END.

R 1,17
: Replaces lines one through 17.

RESTORE

Restore a Previously Saved USL

The RESTORE command is used to copy a previously saved USL (together with any attached files) from the SI file back to the USL file. The file assignments for the USL and any attached files must be the same as when the USL was originally saved with the SAVE command.

If the SI file contains more than one USL, the integer parameter can be used to indicate which USL file to restore. If this parameter is not used, all of the saved files on the SI file are restored.

SED compresses the USL file and any attached files during a RESTORE.

SYNTAX

[.] RES[TORE] [integer]

[integer] (OPTIONAL) Parameter 1 specifies the number of double file marks minus one to advance the SI file before restoring the USL. If not specified, all saved files on the SI file are restored.

Define or Clear Revision String

The REVISION command is used to automatically place a specified character string into every source line that is added or changed with the ADD, REPLACE, CHANGE, MODIFY, OVERLAY, IST, and MST commands. The revision string will be placed beginning in the column specified by the integer parameter. If not specified, the string will begin in column 73.

In sequential mode SED inserts the revision string into the source line just before writing the line to the SO file. In random mode SED inserts the string into the line when the line pointer moves to a new line.

To clear a previous revision string, use the REVISION command without specifying any parameters. This will prevent further insertion of the revision string but will not remove any revision strings already in place.

SYNTAX

[.] REV[ISION] [(string)] [integer]

[(string)] (OPTIONAL) Parameter 1 is optional if you want to clear the current revision string and required if you want to define a revision string. If specified, the character string must be enclosed in any of the valid string delimiters that were listed in Section 3.1.

[integer] (OPTIONAL) Parameter 2 can only be used if a string was specified in Parameter 1. The integer specifies the beginning column number where the string is to be inserted. If not specified, the string will be inserted beginning in column 73.

EXAMPLES

REV "*A1"

Inserts the string *A1 into every source line that is added or modified. The string will be placed in columns 73 through 75.

.REVISION (REV C.1) 68

Inserts the string REV C.1 into every source line that is added or modified. The string will be placed in columns 68 through 74.

REV

Clears the current revision string.

SAVE

Save a USL

The SAVE command copies a USL to the SO file in a special saved format. This command is useful if you want to copy a USL to another device. Once the USL is in saved format, it can only be read by the RESTORE and TEST commands.

When a SAVE is initiated, SED copies all of the source files that are cataloged on the USL file, any attached files, and the USL directory to the SO file. The directory is compressed while being copied so that all directory sectors are at the beginning of the USL file and all source files and attached files are copied at the beginning of the USL. The saved file will then be compressed when it is restored.

The file assignment for SI will be changed during a SAVE.

SYNTAX

[.] SAV[E]

Display Scale with Current Tab Settings

The SCALE command displays a scale numbered from 1 through 79 and prints a T above each column position where a tab is currently set. The display is written to the CO file.

The PTA command can also be used to tell where tabs are set. This command only reports the column positions of the current tab settings without printing the scale.

To clear tabs, use the CLEARTAB command. To set tabs, use the SETTAB command.

SYNTAX

[.] SCA[LE]

EXAMPLES

The output format of an SCA command is shown below. This example shows that tabs are set at columns 7, 21, 36, and 73, which are the default tab settings when you first invoke SED. These are also the default settings for FORTRAN statements.

```
      T  1          2T         3      T  4          5          6          7  T
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

SEARCH

Search for Character Strings

The SEARCH command is used to locate source lines that contain a specific character string. When any matches are found, SED displays those lines on the CO file. After the search, the line pointer is reset to its prior position.

The search can be limited to a specific number of lines from the current line by specifying an integer after the search string. Otherwise the search always begins at the current line and continues to the end of the source program.

If SEARCH is used without any parameters, SED uses the search string that was specified in the most recent SEARCH command and initiates the search from the current line to the end of the file. Any values that were previously specified for the integer parameter are not carried over.

If the range of columns that SED will examine for search strings has been restricted through the use of the LIMIT command and option, SED will not find the string if it appears outside of those columns. Refer to the description of the LIMIT command for more information.

SYNTAX

[.] SEA[RCH] [(string)] [integer]

[(string)] (OPTIONAL) Parameter 1 is optional if you want to search for the character string used in the most recent SEARCH command. Otherwise it specifies the string for which to search, beginning at the current line.

[integer] (OPTIONAL) Parameter 2 initiates the search for the next "integer" lines, beginning at the current line. Note that the current line is not counted in this value, even though SED searches for the string in that line. For example, a value of 6 would cause a search on seven lines (the current line through the next six lines).

EXAMPLES

.SEARCH "NAME"

Searches from the current line through the end of the source file for the string NAME.

SEA :BRX,8: 200

Searches for the string BRX,8 from the current line through the next 200 lines (i.e., the total lines searched would be 201).

SEARCH , 20

Searches for the string that was specified in the most recent SEARCH command. The search begins at the current line and continues through the next 20 lines (i.e., the total lines searched would be 21). Note that one blank space is required between the delimiter (the comma) and the integer parameter.

SEA

Searches for the string that was specified in the most recent SEARCH command, beginning at the current line and continuing through the end of the file. Any integer value that may have been used in the most recent SEARCH command is not carried over.

Generate Four-Digit Sequence Numbers

The SEQUENCE command is used to replace columns 77 through 80 of each source line in a file with a four-digit output sequence number.

If a label is specified, it is placed in columns 73 through 76 of each statement. If a label is not specified, columns 73 through 76 remain unchanged.

The EXSEQUENCE command can be used after a SEQUENCE command to cause the sequence numbers to be five digits instead of four digits. If a label is specified with SEQUENCE and then EXSEQUENCE is used, the label is truncated to three characters and placed in columns 73 through 75.

For more information on the use of sequence numbers, refer to Section 2.12 in Chapter 2 or to the descriptions of the EXSEQUENCE, NOSEQUENCE, and UNSEQUENCE commands in this chapter.

SYNTAX

[.] SEQ[UENCE] [label]

[label] (OPTIONAL) Parameter 1 is a one- to four-character label that will be placed in columns 73 through 76 of each source statement.

EXAMPLES

SEQ CARD

Places the character string CARD in columns 73 through 76 and four-digit sequence numbers in columns 77 through 80 of each source line that is copied to the SO file or every line in the WRK file.

.SEQ,LIBR

Places the character string LIBR in columns 73 through 76 and four-digit sequence numbers in columns 77 through 80 of each source line that is copied to the SO file or every line in the WRK file.

SEQUENCE

Places four-digit sequence numbers in columns 77 through 80 of each source line that is copied to the SO file or every line in the WRK file. Columns 73 through 76 would remain unchanged.

SEQ ABCD
EXSEQ

Places five-digit sequence numbers in columns 76 through 80 and places the string ABC in columns 73 through 75 of every source line that is copied to the SO file or every line in the WRK file.

SETTAB

Set Tabs

The SETTAB command is used to set the column positions for tabs. By default, tabs are set in columns 7, 21, 36, and 73, which are the standard settings for FORTRAN statements. The ALL parameter of SETTAB sets tabs at every column position.

To clear one or more of the current tab settings, use the CLEARTAB command. To display the current tab settings, use the PTA or SCALE command.

Tabs can be used to format input data when entering text for a new source file, adding or replacing records with the ADD or REPLACE commands, or overlaying specific characters with the OVERLAY command. To request a tab, press the CONTROL and I keys simultaneously. The DEFINE command (with the TAB parameter) can be used to redefine the tab character to another key (e.g., a single keyboard character).

For more information on using tabs to format input data, refer to Section 2.3.4 in Chapter 2.

SYNTAX

[.] SET[TAB] { integer [integer] ... | ALL }

Parameter 1 is required and can be one of the following:

integer one or more positive integers that specify the number(s) of the column position(s) where tabs are to be set.

ALL used to set tabs at every column.

EXAMPLES

SETTAB 7,21,36,73
Sets tabs in columns 7, 21, 36, and 73 (standard tab settings for FORTRAN statements).

SET 8,21,36,70
Sets tabs in columns 8, 21, 36, and 70 (standard tab settings for Assembly language statements).

Expand Size of USL Directory

The SIZE command causes SED to update a USL directory regarding information about the size of the file to which logical file USL is assigned. This command should be used when the USL is assigned to a MAX IV or MAX 32 File Manager file that has been expanded since it was initialized or since the last SIZE command was used.

SYNTAX

[.] SIZ[E]

SKIPTO

Move Line Pointer

The SKIPTO command has the same effect as entering a line number as a command itself; that is, it moves the line pointer from its current position to the specified line within the source file. That line then becomes the new current line. With the SKIPTO command, you can reference a line by its line number, by a character string within the line, or by an offset from the current line.

The SKIPTO command is similar in function to the GOTO command except that in sequential mode, source lines that are bypassed are **not** copied to the SO file. With GOTO, any lines that are bypassed in sequential mode are copied from the SI file to the SO file.

In random mode you can skip to a line that is before or after the current line. In sequential mode you can only skip to a line that is located after the current line on the SI file.

If the SLR parameter is a character string reference and column limits have been imposed by a previous LIMIT command, those limits are in effect when SED searches for the character string.

To initiate a backward search in random mode using a character string reference, precede the string delimiter with a minus sign as shown in the last two examples. If the SLR parameter is a line number and that line is somewhere before the current line, you do not have to use the minus sign. For example, SKIPTO 4 will move the pointer to line 4 even if the current line is line 20.

After a SKIPTO using a character string reference, you can then use SKIPTO without a parameter to skip to the next or previous occurrence of that string. The search starts at the current line plus one if the most recent SKIPTO used a forward character string reference. If the SLR parameter from the most recent SKIPTO used a backward character string reference, the search starts at the current line minus one and goes backward.

Note that when a character string is used for the SLR parameter (either forward or backward), the search for that string always starts at the current line. Therefore if the string appears on the current line, the line pointer will remain on that line. Entering another SKIPTO without an SLR parameter will then move the pointer to the next or previous occurrence of that string as described above. When the string can no longer be found, SED will return the message SOURCE LINE NOT FOUND.

If the NOQUICK option or command is in effect, SED displays the current line on the CO file after a SKIPTO command.

In random mode you can also move the line pointer forward one line at a time by simply pressing the carriage return.

SYNTAX

[.] SKIPTO [[-]SLR]
SKIP
SKI

[SLR] (OPTIONAL) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3. It references the line that you want to skip to. If not specified, SED uses the value from the previous SKIPTO or GOTO command (whichever was used most recently).

EXAMPLES

SKIPTO 1{SPOT}

Moves the pointer to the first line encountered that contains the string SPOT starting in column one. If it is found on the current line, the pointer will remain on the current line. Entering another SKIPTO without a parameter would then move the pointer to the next occurrence of the string SPOT in column one.

SKIP -/HEX/

Searches backward for the line that contains the string HEX. If that string appears on the current line, the pointer would remain on the current line. Entering another SKIPTO without a parameter would then move the pointer to the previous occurrence of the string. Because this is a backward search, this example would only be valid in random mode.

SKIPTO -/ALPHA/-1

Searches backward for the line that contains the string ALPHA and then sets the pointer to one line before that. If the string appears on the current line, the pointer will be set to one line before the current line. Entering another SKIPTO without a parameter would cause SED to search for the previous occurrence of the string and set the pointer to one line before that. Because this is a backward search, this example would only be valid in random mode.

.SKI 17

Moves the pointer to line 17.

SKIP C+52

Moves the pointer 52 lines beyond its current position.

SOURCE

Set Source Format for LO Listings

The SOURCE command sets the format for all subsequent listings that are written to the LO file. In source format, listings are printed with headers, titles, and source line numbers. The opposite of source format is text format, which is requested by the TEXT command. In text format, listings are printed without headers, titles, and source line numbers.

SOURCE is the default mode when SED is first invoked.

Figures 3-11 and 3-12 are sample listings of the same source file. In Figure 3-11, the SOURCE command is in effect. In Figure 3-12, the TEXT command is in effect.

SYNTAX

[.] SOU[RCE]

```
POS ARGMIS
SOURCE
LIST
*G.Ø  MODCOMP SOURCE EDITOR      DATE  Ø9/19/85  16:47:24        PAGE   1


  1  * ASSEMBLER CONDITIONAL DIRECTIVES
  2  *
  3  TEST  MAC
  4        IFP           %1,A
  5        DFC           5
  6  A     IFM           %1,DONE
  7        DFC           %1
  8  DONE  AOP
  9        EMP
 10  *
 11  * SOME CALLS TO THE MACRO:
 12  *
 13        TEST
 14  *
 15        TEST,Ø
 16  *
 17        TEST          9
 18        END
```

Figure 3-11. LIST in Source Format

```
POS ARGMIS
TEXT
LIST
* ASSEMBLER CONDITIONAL DIRECTIVES
*
TEST  MAC
      IFP           %1,A
      DFC           5
A     IFM           %1,DONE
      DFC           %1
DONE  AOP
      EMP
*
* SOME CALLS TO THE MACRO:
*
      TEST
*
      TEST,Ø
*
      TEST          9
      END
```

Figure 3-12. LIST in Text Format

SPLIT

Split a Source Line into Two Lines

The SPLIT command is used to split a source line into two separate lines.

When this command is used, SED displays the specified line on the CO file. To split that line, enter an uparrow (^) below the character that is to appear in the first column of the second line.

SYNTAX

[.] SPL[IT] [SLR]

- [SLR] (OPTIONAL) Parameter 1 is a source line reference as previously described in Sections 2.4.2.1 through 2.4.2.3 that references the line you want to split. If not specified, SED displays the current line so that it can be split.

EXAMPLES

SPL 10
Displays line ten so that it can be split.

SPL C+2
Displays the line located two lines after the current line so that it can be split.

.SPL 4"ABC"
Displays the first line encountered with the character string ABC starting in column four so that it can be split.

Squeeze a USL Directory

The SQUEEZE command compresses all directory sectors and source files on the USL file and any attached files to the beginning of their disc partitions. You should periodically request a directory listing with the LDIRECTORY command to determine if a USL needs to be squeezed. If it does, the message "NEEDS TO BE COMPRESSED" appears at the end of the listing as shown in the sample output under the LDIRECTORY command (Figure 3-3).

A SQUEEZE may be either a two or four-pass operation:

Pass 1	SED copies the USL file and any attached files to the SO file and displays the message "COPY COMPLETED" on the CO file.
Pass 2	(OPTIONAL) SED checks the SO file for data integrity.
Pass 3	SED copies the SO file (that now contains the USL and any attached files) back to the USL file and then lists the directory on the LO file.
Pass 4	(OPTIONAL) SED checks each source file in the USL for data integrity as it lists the directory.

If the NOTEST parameter is specified, only passes 1 and 3 are executed.

SYNTAX

```
[.] SQU[EEZE] [NOTEST]
              [NOT]
```

TEST

Test Data Integrity of Programs on SI File

The TEST command checks the data integrity of source files on the SI file. The files must be in saved format (i.e., as produced by the SAVE command).

SYNTAX

[.] TES[T]

TEXT

Set Text Format for LO Listings

The TEXT command sets the format for all subsequent listings that are written to the LO file. In text format, listings are printed without headers, titles, and source line numbers. The opposite of text format is source format, which is requested by the SOURCE command. In source format, listings are printed with headers, titles, and source line numbers.

SOURCE is the default mode when SED is first invoked.

See Figures 3-11 and 3-12 under the SOURCE command for sample output of listings in both source and text format.

SYNTAX

[.] TEX[T]

Copy Source Records to SO File in Compressed Format

The TRANSFER command is used to copy compressed or uncompressed records to the SO file in compressed format. In random mode TRANSFER always copies the entire WRK file to the SO file. In sequential mode TRANSFER copies one or more programs from the SI file to the SO file as indicated by the optional integer parameter.

In sequential mode source programs are copied from the current position of the SI file to the current position of the SO file. File marks are copied to the SO file when they appear on the SI file unless the NOFILE command is in effect. To copy more than one program at a time in sequential mode when each program is separated by a file mark, use the integer parameter. This copies programs until the specified number of file marks have been bypassed. A double file mark terminates the transfer even if the specified number of file marks have not been encountered. If an integer is not specified, SED copies the program from the current position of the SI file until it encounters a file mark.

TRANSFER always uses $$ as its file mark. Any other file marks that may have been defined with the MARK command have no effect on the TRANSFER command. In addition, the COMPRESS, NOCOMPRESS, SEQUENCE, EXSEQUENCE, and UNSEQUENCE commands have no effect on the TRANSFER command.

SYNTAX

Random Mode:

[.] TRA[NSFER]

Sequential Mode:

[.] TRA[NSFER] [integer]

[integer] (OPTIONAL) Parameter 1 can only be used in sequential mode. If used, it specifies the number of programs you want to copy from the SI file to the SO file (i.e., number of file marks to bypass). If not specified, SED copies the current program on the SI file until it reaches the first end-of-file mark.

EXAMPLES

TRA 3

Copies three programs from the current position of the SI file to the current position of the SO file. Because an integer is specified, this example would only be valid in sequential mode.

TRANSFER

In random mode, this example would copy the WRK file to the SO file. In sequential mode it would copy the program from the current position of the SI file to the SO file until a file mark is encountered.

TWIN

Assign Alias Name to a Source File

The TWIN command assigns an alias name to a source file by creating another entry for it in the USL directory. The file can then be referenced by its original name or the alias name.

SYNTAX

[.] TWI[N] name1 name2

name1 (REQUIRED) Parameter 1 is the name of the cataloged file to be given an alias name.

name2 (REQUIRED) Parameter 2 is the alias name.

EXAMPLES

.TWIN ABC,DEFG
The source file ABC can now be referenced as either ABC or DEFG.

TWI HOST,MASTER
The source file HOST can now be referenced as either HOST or MASTER.

UNCHECK

Suppress Checking of Sequence Numbers

The UNCHECK command suppresses the checking of sequence numbers in columns 77 through 80 of each source line. This command would be used to suppress sequence checking that was initiated by a CHECK command.

UNCHECK is the default mode when SED is first invoked.

SYNTAX

[.] UNC[HECK]

UNDO

Reverse the Effect of the Most Recent CHANGE Command

The UNDO command is used to override the effect of the most recent CHANGE command. In effect, SED takes the string1 and string2 parameters from the most recent CHANGE and simulates a CHANGE command with string2 as the first string and string1 as the second string.

In order to use the UNDO command, the pointer must be positioned on the line on which the change was originally made.

SYNTAX

[.] UNDO
UND
U

UNSEQUENCE

Insert Blanks in Columns 73 - 80

The UNSEQUENCE command causes SED to insert blank spaces in columns 73 through 80 of each source statement that is written to the SO file.

If you have also used a SEQUENCE command, SED inserts the four-digit sequence number in columns 77 through 80. However, if a label parameter was not specified in the SEQUENCE command, columns 73 through 76 of each source statement would end up as blank spaces because UNSEQUENCE would first have placed blanks in columns 73 through 80.

For more information on the use of sequence numbers, refer to Section 2.12 in Chapter 2 and the descriptions of the SEQUENCE, EXSEQUENCE, and NOSEQUENCE commands in this chapter.

SYNTAX

[.] UNS[EQUENCE]

USE

Place an Identification in the USL Directory

The USE command records a specified character string in the directory of the USL file. The string will be displayed in the I.D. field at the top of a directory listing. The string overrides the identification string that may have been specified when the directory was first initialized with the INITIALIZE command.

See Figure 3-3 under the LDIRECTORY command for a sample directory listing showing the I.D. field.

SYNTAX

[.] USE id

id (REQUIRED) Parameter 1 is a one- to eight-character string.

EXAMPLES

USE USL1
Places the identifier USL1 in the directory of the USL file.

USE MACROS
Places the identifier MACROS in the directory of the USL file.

Verify Checksum Values

The VERIFY command causes SED to read source records from the SI file and verify checksum values. If a checksum error is found, SED displays an error message.

By default, verification is performed until SED encounters a file mark or double file mark on the SI file. To verify more than one program at a time, use the integer parameter. This verifies source programs until SED bypasses that number file marks. A double file mark terminates the verification even if the specified number of file marks have not been encountered.

SYNTAX

[.] VER[IFY] [integer]

[integer] (OPTIONAL) Parameter 1 is the number of file marks to bypass before verification is terminated. If not specified, the default is to continue verifying checksum values from the SI file until one file mark or a double file mark is encountered.

EXAMPLES

.VERIFY 3

Verifies checksum values from the SI file until SED encounters three file marks.

VER

Verifies checksum values from the SI file until the first file mark or a double file mark is encountered.

Modify Output Listings

The XLN command is used prior to any command that generates listings of source files to the LO file (e.g., LIST and LAE with the LIST parameter). If XLN is used without any parameters, the listings on the LO file will only contain those source lines with the following character sequences in them:

Special Identifier	Meanings and/or Position
((	Macro insert file identification, starting in column one
PGM name	Starting in other than column one
*CAT NUM	Where "*" is in column one and "CAT NUM" starts in any other column (used for MCS catalog numbers)
CCAT NUM	Same as *CAT NUM
CCAT NO	Same as *CAT NUM
CCAT #	Same as *CAT NUM
END	Starting in other than column one
BLOCK DATA	Starting in other than column one
PROGRAM	Starting in other than column one
SUBROUTINE	Starting in other than column one
OVERLAY	Starting in other than column one
FUNCTION	Starting in other than column one
INTEGER FUNCTION	Starting in other than column one
LOGICAL FUNCTION	Starting in other than column one
REAL FUNCTION	Starting in other than column one
DOUBLE PRECISION FUNCTION	Starting in other than column one
COMPLEX FUNCTION	Starting in other than column one
TTL	Starting in other than column one
STL	Starting in other than column one
INT	Starting in other than column one
EXT	Starting in other than column one
INS	Starting in other than column one
INS@	Starting in other than column one
INS*	Starting in other than column one

XLN can also take as parameters one or more user-defined strings. If specified, the listings on the LO file will contain all source lines with the character sequences listed on the previous page and those lines with the user-defined strings in them.

The LNAME, LNS, and LNT commands are almost identical to XLN except the list of predefined character sequences differs slightly. For an example of the effect these commands have on a LIST and LAE command, see Figures 3-5, 3-6, 3-7, and 3-8 under the LNAME/LNS/LNT command description in this chapter.

To override the effect of an XLN command, use the LALL command. LALL is the default mode when SED is first invoked.

SYNTAX

[.] XLN [(string1)] [(string2)] ...

The optional character strings must each be enclosed in any of the valid string delimiters that were listed in Section 3.1. If no strings are specified, any command that lists source files to the LO file will only list the lines that contain any of the predefined character sequences.

EXAMPLES

.XLN

Listings of source files to the LO file will contain only those lines with the predefined sequences of characters in them.

XLN "BRU",:HOP:,/BRX/

Listings of source files to the LO file will only contain those lines with the predefined sequences of characters in them as well as the lines that contain the strings BRU, HOP, and BRX.

XRN
YON

XRN (Random Mode Only)

Exit Random Mode

The XRN command is used to terminate random mode and return to sequential mode. Another way to exit random mode is to use the RANDOM EXIT command.

SYNTAX

[.] XRN

YON (Random Mode Only)

Set Current Position in WRK File

The YON command can only be used in random mode. It causes SED to set the current position of the WRK file to the position that was saved as a result of the last HITHER command.

Refer to the description of the HITHER command in this chapter for more information and a usage example of HITHER and YON.

SYNTAX

[.] Y[ON]

APPENDIX A
ERROR REPORTING

ERROR MESSAGES AND RECOVERY

SED writes all error and warning messages to the device assigned to logical file CO. If the NOHOLD option is in effect and the CI file is assigned to your terminal, you can reenter the command correctly after an error message. If the HOLD option is set and the CI file is not assigned to your terminal, SED will enter a hold state after printing an error message. To resume execution, activate the Operator Communication (OC) task and use the RESUME directive (/R). This will take SED out of hold so that you can continue normal operation.

The remainder of this appendix lists the SED error messages, gives the cause of the error, and recommends a course of action for correcting the error.

! CONTROL STATEMENT ERROR

A command was entered incorrectly. This error could occur for one of many reasons: a misspelled command name, invalid parameters, a character string that was not enclosed in valid delimiters, or a command that was entered out of sequence in sequential mode. Check the proper syntax and use of the command in Chapter 3.

CANNOT MODIFY SI FILE ASSIGNMENT

The SI file is assigned to a MAX IV or MAX 32 File Manager file in a manner that prohibits it from being reassigned to an attached file.

CANNOT SAVE A NULL PROGRAM

The SAVE and SQUEEZE commands cannot be used to save or squeeze USL files that contain null programs (end-of-file mark only). Any null programs should first be removed witht the REMOVE command before using the SAVE or SQUEEZE commands.

CHECKSUM ERROR

The checksum value of an input record is not equal to its associated value in the record. If SED does not hold, you can do one of the following:

- Enter RSM to continue reading records and checking for data integrity.
- Enter RSM NO to continue reading records without checking for data integrity (until an end-of-file is encountered).
- Enter RSM STOP to stop reading records. In this case the SI file should be reassigned.
- Enter any other command to abort the current command and proceed. In this case the SI file should be reassigned.

If SED does hold after a checksum error, activate the OC task, enter the single character / and follow one of the first three steps above, using the character R in place of the characters RSM.

DIRECTORY IN ERROR

A command that requires a directory to be present on the USL file has been used and an invalid directory has been found. If the command was a disc-to-disc COPY ALL, the SI and SO files must contain a directory. If the command was a disc-to-tape COPY ALL, the SI file must contain a directory.

END OF MEDIA

A read was attempted at the end of a tape or disc. Reposition the appropriate logical file (e.g., with REWIND) and reenter the command.

FILE IS NOT ATTACHED

The space on the specified file could not be detached because the file was not attached.

FILE MANAGER FILE MAY NOT BE ATTACHED

An ATTACH command has been used and the specified file name is a MAX IV or MAX 32 File Manager file. A File Manager file cannot be attached.

FILE MUST BE DISC

The file referenced by the command must be a disc file.

FILE MUST BE DISC OR MAGNETIC TAPE

For the LDIRECTORY command, the USL file must be disc or magnetic tape. For other commands, this message applies to the SI file.

FILE PREVIOUSLY ATTACHED

The space on the file specified by the ATTACH command was already available to the USL directory.

ILLEGAL COMBINATION OF MARKS

The specified program and subprogram marks cannot be used together. Redefine the marks to some other combination of characters using the MARK command.

ILLEGAL FOR MODE OF OPERATION

The specified command cannot be used in the present mode of operation. You may have tried to exit random mode and you were already in sequential mode. Some SED commands automatically exit random mode after they are processed. Make sure that the command you are trying to enter can be used in the mode you are currently in.

INTERNAL ERROR

SED has detected an abnormal situation (not a user error). A memory dump of the processor will be generated. You should report this problem to MODCOMP by filling out a Software Problem Report (SPR), attaching the memory dump and as much supporting evidence as possible to recreate the sequence of events that led to the error.

INVALID FILE OPERATION

An invalid file name was specified. Reenter the command using the correct file name.

INVALID INPUT RECORD

A compressed source record with an invalid byte count has been read.

INVALID SAVE FORMAT

A RESTORE command has been used and the SI file is assigned to a file that is not in saved format (i.e., the format produced by the SAVE command). Reassign the SI file to a file that is in saved format and then restore it.

MACROS NESTED TOO DEEP

A macro called another macro and the macros were already nested four levels deep. Remove the macro definition and recreate it with four or less nesting levels.

NAME PREVIOUSLY USED:name

You tried to catalog a source program with the same name as an existing program. If you are trying to replace the previous version with the newer version, use the RECATALOG command instead of CATALOG. Otherwise catalog the program with a unique name.

NO CHANGE

The characters specified by the CHANGE command were not found and no lines were changed. Check to make sure that you spelled the string correctly. If you did not specify to make the change to more than one line, SED only looks for the string on the current line and it was not found on that line. If you have protected certain columns from being modified through the LIMIT command and option, then data in those columns cannot be modified by a CHANGE command.

NO DIRECTORY

The specified command requires that a directory be present on the USL file. If the command was a disc-to-disc COPY ALL, the SI and SO files must contain a directory. For other commands, this error message applies to the SI file.

NO ENDM FOR END-OF-MACRO

Macro stack space has been exhausted before an ENDMACRO command to end the current macro was processed. Remove any macros that you no longer need and try to execute the macro again. The number of macros that can be defined at any one time is installation-dependent and can be changed by the system administrator.

NO FILE TO RECOVER

The last program written on the WRK file in random mode is no longer available.

NOREWIND CAN ONLY APPLY TO MAGNETIC TAPE

A command has been used with the NOREWIND parameter and neither the SI nor SO files are assigned to a magnetic tape. Respecify the command again without the NOREWIND parameter.

PARTITION SIZE NOT COMPATIBLE

A RESTORE command was used and the USL file is assigned to a partition that is smaller than the one that was saved. Change the file assignments so that they are the same as when the USL was saved with the SAVE command.

PROGRAMS ON THE FILE!

An attached file could not be detached because it still contains source programs. These programs must first be removed with the REMOVE or RAE command. After the programs are removed, the attached file can then be detached.

PROGRAM NOT PRESENT

The specified program (or subprogram) could not be located on the SI file. This usually occurs because the SI file is not assigned to the USL or because you misspelled the program name.

RECORD NOT ACCESSIBLE

The OVERLAY command could not be performed because no source line was currently accessible. Use GOTO, SKIPTO, or some other SED command to position to the appropriate line and then reenter the OVERLAY command.

SECTOR WORD SIZE NOT COMPATIBLE

For a SAVE command, this means that the SO file has smaller sectors than the file trying to be saved. Reassign the SO file to a device with the appropriate size. For a RESTORE command, this means that the sector size of the file to which a RESTORE is being done is not equal to the sector size of the file from which a SAVE was done. Change the file assignments so that they are the same as when the file was originally saved.

SEQUENCE CHECK ERROR

The sequence number in columns 77 through 80 of this record is less than the sequence number of the previous record. If SED does not hold, you can do one of the following:

- Enter RSM to continue reading and checking sequence numbers, using the value of the sequence number from the last source line that was in sequence.
- Enter RSM LOW to continue reading and checking sequence numbers, resetting the value of the sequence counter to zero.
- Enter RSM NO to continue reading records without checking the order of sequence numbers.
- Enter any other command to abort the current command and proceed. In this case the SI file should be reassigned.

If SED does hold, activate the OC task, enter the single character / and follow one of the first three steps, using the character R in place of the characters RSM.

SO FILE IS ASSIGNED TO A USL DIRECTORY

A COPY, SAVE, or SQUEEZE command has been used and logical file SO is assigned to a disc partition with a USL directory. If the HOLD option is set, SED will hold and you can then enter /R to continue or /A to abort SED. If the NOHOLD option is set, SED will NOT issue this error message but will continue with the copy or save operation.

SOURCE LINE NOT FOUND

The source line specified in the command could not be found. If you used a character string reference, make sure that the string is enclosed in valid delimiters and that you spelled the string correctly. This error may also be caused by a previous use of the LABEL command, which restricts searches for character strings to certain columns. To override those limits, use the LABEL command without specifying any parameters. Another possible cause is that the command searched for the string beginning at the current line and the string is located somewhere before the current line.

SPACE IS INSUFFICIENT

The space on the file to which source lines are being written has been exhausted and more source lines remain to be written.

USER MUST REASSIGN SI FILE

A POSITION command has been used and a previous POSITION command has caused SI to be assigned to an attached file and the file containing the directory was a MAX IV or MAX 32 File Manager file. Reassign SI back to the file containing the directory (using the ASSIGN command) and then use the POSITION command again.

WARNING MESSAGES

The following messages are warning messages indicating a potential problem. Even though you are warned, SED will still process the command. You should check the results of the operation to make sure that it is what you intended to do.

PROGRAM HAD NOT BEEN PREVIOUSLY CATALOGED:name

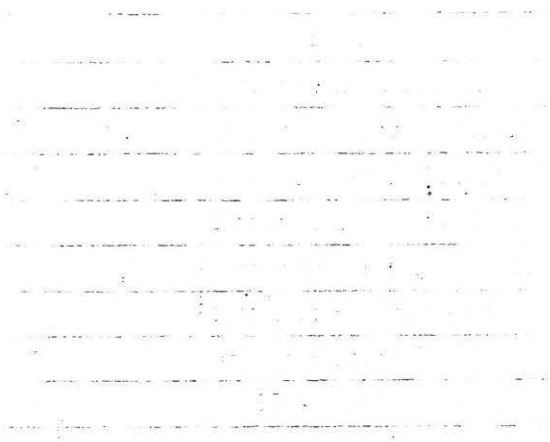
A RECATALOG command has been used and the program was not originally cataloged. SED will catalog the program with the name used in the RECATALOG command.

WARNING ! : TAPE STILL ON FIRST FILE

A COPY ALL NOREWIND command has been used and the tape is at the beginning-of-tape (BOT). The programs being copied to the tape will overwrite whatever is currently on the tape.

INITIALIZATION REQUESTED

An initialization of the USL file has been requested. If SED is resumed, the initialization will take place and all previous data on the USL file will be lost.



APPENDIX B
DIRECTORY ORGANIZATION AND FILE FORMATS

This appendix describes and illustrates the internal format of entries for a USL directory, attached logical files, cataloged source programs, the USL file, the WRK file, saved files, and compressed records.

Directory Entries

The first sector of the USL partition is always the first directory sector. Each directory sector consists of two words that are pointers to the previous and the next directory sectors, followed by a number of nine-word directory entries. The number of entries per sector depends on the size of the sector: a 128-word sector holds 14 entries and a 100-word sector holds 10 entries.

The first word of each directory sector is the file position index (FPI) of the previous directory sector; the second word of each directory sector is the FPI of the next sector. The first word of the first directory sector is #FFFF; the second word (forward chain pointer) of the last directory sector is also #FFFF.

Directory sectors are created dynamically as needed by SED and thus can be scattered throughout the USL. For efficient use of disc space, it is recommended that the directory be contiguous at the beginning of the USL. This can be achieved either by squeezing the USL or by initializing the USL with enough entries.

The first entry in the directory is for the directory itself and has the following format:

Word:

0		Number of Directory Entries per Sector
1		Number of Entries in Use
2		Number of Entries Not in Use
3	X	Number of Source Programs on the USL File
4		Characters 1, 2, and 3 of USL Identifier
5		Characters 4, 5, and 6 of USL Identifier
6		Characters 7 and 8 of USL Identifier
7		Month/Day Last Modified
8		Year Last Modified

The first bit of relative word 3 is ON if the USL file has been attached and OFF if the USL file has not been attached.

Attached File Entries

The remaining entries in the directory are for attached logical files and the source programs that are cataloged on them. All source programs that are cataloged on the same logical file have their directory entries together. Immediately preceding the directory entry is a file entry for the logical file on which the programs reside. The format of the entry for an attached logical file is as follows:

Word:

0	#FEFE - to show File Entry
1	Logical File Name in CAN-Code
2	Number of Sectors in File Partition
3	Sector Size of Attached Logical File
4	0
5	0
6	0
7	0
8	0

The file entries appear in reverse order of attachment, with the entry for the USL file last. Word 1 (logical file name) of the file entry for the USL is binary zero to indicate that this is the USL.

Source Program Entries

The entries for the source programs on each logical file then follow the attached file entries. These program entries are in the same order as the programs actually reside on the attached logical file and have the following format:

Word:

0	Characters 1, 2, and 3 of Alias Name in CAN-Code
1	Characters 4, 5, and 6 of Alias Name in CAN-Code
2	Characters 7 and 8 of Alias Name in CAN-Code
3	Unused (7 bits) Month/Day of Catalog
4	Year of Catalog
5	Time (in Minutes) of Catalog
6	Relative Disc Sector Address of Program
7	Number of Sectors Occupied by Program
8	Number of Words of Data in Last Sector of Program

Words 3 through 5 contain the date and time that the program was cataloged if the DATE option was set. If the PGM option was set, these words contain the name found on the first PGM or PROGRAM statment within the program (in CAN-code format). If an identification string was specified with the CATALOG command when the source program was cataloged, then words 3 through 5 contain that string (also in CAN-code format). Specifying an identifier takes precedence over the DATE and PGM options. Whatever name is stored in words 3 through 5 will be in the same format as the name in words 0 through 2 (i.e., CAN-coded).

USL File Entries

To maintain consistency, the entries for the USL logical file also contain entries for each directory sector. The format of these entries are:

Word:

0	0
1	0
2	0
3	0
4	0
5	0
6	Relative Disc Sector Address of Directory Sector
7	Number of Sectors Occupied by Directory Sector=1
8	0

The first word in the last directory entry contains the value #FFFF. The remaining words in the last entry, and all subsequent entries, are not used.

WRK File Entries

The WRK file is only used in random mode. The first sector of the WRK file has the following format:

Word:

0	Pointer to First Source Record
1	Pointer to Last Source Record
2	Number of Records per Sector
3	Total Number of Source Records in File
4	Pointer to Last Available Slot
5	Pointer to Next Unused Slot
6	Pointer to Reclaimed Slot Chain
7	Random Record Flag
8	#ABCD - Verify Character for Recovery

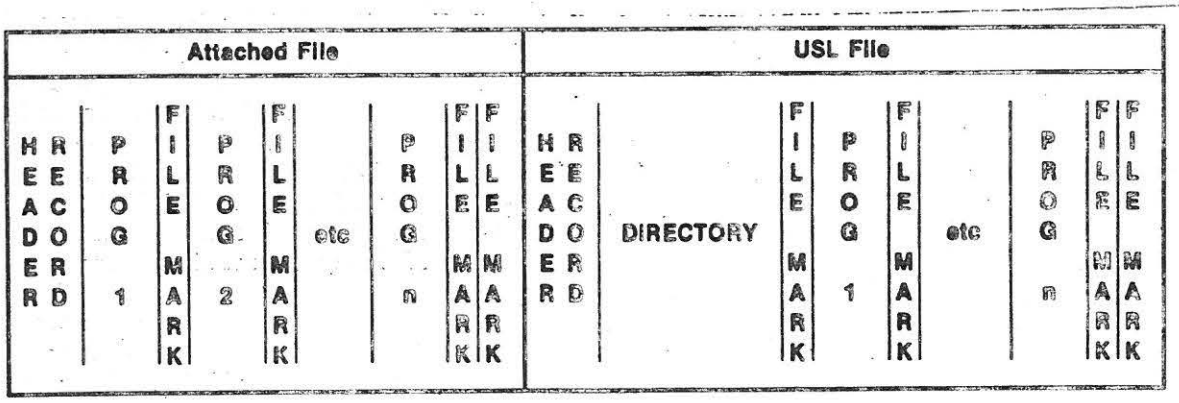
The remaining sectors contain the actual source records of each program.

All source records on the WRK file are in uncompressed format. If a sector is 100 words, each sector contains two records. If a sector is 128 words, then each sector contains three source records. Each record is preceded by two words: the first points to the previous source record, the second points to the next source record. A pointer of #8000 indicates that the record is at the end of the chain.

The pointer for the first logical record has a value of #0003. This allows SED to quickly calculate which sector each record resides on by "record pointer modulo 3."

Saved File Entries

Saved files generated by the SAVE command have the following format:



The first record contains the name of the attached file and the word size of its sectors. Each source program is separated by a file mark, with an additional file mark after the last program. If the file name is USL, the header record is followed by the USL directory. Again, each source program is separated by a file mark, with an additional file mark after the last program.

Compressed Format Entries

Compressed ASCII is a mechanism intended to conserve disc space when storing or cataloging source programs. Compressed format records are generated when the COMPRESS command is in effect.

Each sector in compressed format has the following format:

Word:

0	#03	Checksum
1	Byte Count	
2	Compressed ASCII Records	
•		
•		
•		
x		

(where x is the last word of either a 256-byte or 200-byte sector). The byte count is the total number of valid bytes in the sector and is usually the same as the number of bytes in a sector (#100 or #C8). However, when the sector is the last one of a program, the byte count will often be less. The byte count includes the four-byte header overhead. The checksum is computed by summing (modulo 256) every other byte (including the other three header bytes) in the sector with the physical record count and negating it (i.e., all bytes in the sector + record sequence count = zero). The physical record count is the number of sectors read or written.

Each logical record, or source line, is stored in a compressed ASCII record. In this format, if a sequence of three or more of the same character occurs, the sequence is represented by the character followed by the negative of the repeat count. The count is negative (the most significant bit is ON) so that it can be readily distinguished from the seven-bit ASCII character. Trailing spaces are ignored and the last nonspace character is followed by a null (#00). The only exception to this is if the record contains exactly 80 characters with no trailing spaces. In this case there is no delimiter; the compression and decompression routines keep a byte count.

INDEX

ADD Command, 2-16, 3-5
Adding Records to a Source Program, 2-16, 3-5
Alias Program Names, 2-27, 3-100
Assembler Programs (Removing Comments from), 3-68
ATTACH Command, 2-40, 3-7
Attached Files
    Detaching Attached Files, 2-40, 3-19
    Internal Format on Disc, B-2
    Renaming, 3-80

BACKWARD Command, 2-14, 3-8
Brackets
    Used for Command Abbreviations, 3-1
    Used for Optional Parameters, 2-2, 3-1

CAE Command, 3-9
CATALOG Command, 3-10
Cataloging Programs, 2-4, 3-10
CHANGE Command, 2-20, 3-11
    Undoing the Most Recent Change, 3-101
Character Strings
    As Source Line References, 2-11
    Delimiters for Character Strings, 2-2, 3-1
    Limiting Searches to Specific Columns, 2-12, 3-33
    Searching for Specific Strings, 3-88
CHECK Command, 2-36, 3-103
Checksum
    Checksum Errors, 2-37
    Verifying Checksum Values, 3-103
CLEARTAB Command, 2-5, 3-13
Commands
    Abbreviations, 3-1
    Format for Entering, 2-2
    SED/Job Control, 3-3
    Summary of All SED Commands, 1-2
COMPRESS Command, 3-14
Compressed Records, 2-30, 3-14, 3-99
    Internal Format, B-5
CONTINUE Command, 2-33, 3-14
Control Characters, 3-71
Copying Programs, 2-30
    COPY Command, 2-30, 3-15
    Copying a USL Disc-to-Disc, 2-32, 3-15
    Copying with the TRANSFER Command, 2-31, 3-99
    Disc-to-Tape or Tape-to-Disc Copy, 2-32
Current Line, 2-13
    Displaying the Current Line 2-14, 3-43, 3-60

DEFINE Command, 2-5, 2-18, 3-17
Defining Strings, 2-22, 3-21
    Revision Strings, 3-85

DELETE Command, 2-17, 3-18
Deleting Programs, 2-27, 3-77, 3-81
Deleting Records from a Program, 2-17, 3-18
Deleting Strings from Source Lines, 2-23, 3-56
DETACH Command, 2-40, 3-19
Detaching Logical Files, 2-40, 3-19
Directories
    Compressing, 2-38, 2-40, 3-97
    Creating a USL Directory, 2-3, 3-30
    Expanding the Size of a USL Directory, 3-91
    Internal Format of Entries on Disc, B-1
    Listing Program Names from a Directory, 2-28, 3-34, 3-36, 3-38
    Listing Statistics for Sector Use, 2-38, 3-40
    USL Identification, 3-30, 3-102
DISPLAY Command, 2-25, 3-20
Displaying Source Programs by Screenful, 2-25, 3-20
DST Command, 2-22, 3-21
DUPLICATE Command, 2-19, 3-22
Duplicating Records, 2-19, 3-22

End-of-File Marks
    Defining, 2-28, 2-52
    Suppressing File Marks on the SO File, 3-59
    Writing File Marks to the SO File, 3-25
ENDMACRO Command, 2-33, 3-23
Error Messages, Appendix A
Errors, 1-11, Appendix A
EXIT Command, 1-10, 3-23
Exiting Random Mode, 3-78, 3-106
Exiting SED, 1-10, 3-23
EXSEQUENCE Command, 2-35, 3-24, 3-59

FILE Command, 2-30, 3-25
File Marks (See End-of-File Marks)
FORWARD Command, 2-14, 3-25

GOTO Command, 2-13, 3-26

HITHER Command, 3-28
HOME Command, 3-29

IGE Command, 3-57
INSERT Command, 2-34, 3-31
Insert Files, 2-34
    With ADD Command, 2-34, 3-5
    With NRE, IRE, NPU, IPU Commands, 3-74
Inserting New Lines into a Program, 2-16, 3-5
Inserting Strings into Source Lines, 2-22, 3-32
Invoking SED, 1-8
IPU Command, 3-62
IRE Command, 3-63
IST Command, 2-22, 3-32

Job Control Commands, 3-3

LABEL Command, 2-12, 3-33
Labels
    Inserting Labels into Source Lines, 2-35, 3-32
    Protecting the Label Field, 2-36, 3-42
LAE Command, 3-34
LALL Command, 3-35, 3-45
LDE Command, 3-36
LDIRECTORY Command, 2-38, 3-38
LDS Command, 3-40
LEOF Command, 3-41
LIMIT Command, 2-36, 3-42
LIMIT Option, 1-9, 2-36, 3-42, 3-64
Line Number References, 2-10
Line Pointer, 2-13
    Automatically Displaying the Current Line, 2-14, 3-60
    Moving the Location of the Line Pointer, 2-13
    Restoring Current Position in the WRK File, 3-28, 3-106
    Saving the Current Position in the WRK File, 3-28
LINENUMBER Command, 3-43
LIST Command, 2-25, 3-44
List Directory Entries, 3-36
List Directory Statistics, 3-40
List Source Programs, 2-24
    In Source Format, 3-94
    In Text Format, 3-98
LNAME Command, 2-26, 3-45
LNS Command, 2-26, 3-45
LNT Command, 2-26, 3-45
Logical Files, 1-1, 1-10
LST Command, 2-26, 3-48

MACRO Command, 2-33, 3-50
Macros
    Continuing the Execution of, 2-33, 3-14
    Defining, 2-33, 3-50
    Ending a Macro Definition, 2-33, 3-23
    Listing the Definitions of, 2-33, 3-50
    Removing, 2-33, 3-50
MARK Command, 2-28, 3-52
Messages to CO File, 3-3, 3-70
Modes of Operation, 1-1, 2-6
    Exiting Random Mode, 3-78, 3-106
    Entering Random Mode, 3-78
    Sequential Referencing, 2-15
MODIFY Command, 2-21, 3-53
Modifying Characters in a Source Line, 2-19
    With the CHANGE Command, 2-20, 3-11
    With the MODIFY Command, 2-21, 3-53
    With the OVERLAY Command, 2-20, 3-66
Modifying Output Listings
    With the LNAME, LNS, LNT Commands, 2-26, 3-45
    With the LST Command, 2-26, 3-48
    With the XLN Command, 2-26, 3-104
MOVE Command, 2-19, 3-55

Moving Records within a Program, 2-19, 3-55
MST Command, 2-23, 3-56

NGE Command, 3-57
NOCOMPRESS Command, 3-58
NOFILE Command, 2-30, 3-59
NOPACK Command, 3-59
NOQUICK Command, 2-14, 3-60
NOQUICK Option, 1-10, 3-60, 3-65
NOSEQUENCE Command, 2-35, 3-61
NPU Command, 3-62
NRE Command, 3-63

OPTION Command, 3-64
Options, 1-8, 3-64
OVERLAY Command, 2-20, 3-66

PACK Command, 3-68
PAUSE Command, 3-70
PC Command, 3-71
POSITION Command, 2-16, 3-72
PRF Command, 3-74
PRINT Command, 2-24, 3-75
Printing Source Programs, 2-24, 3-75
Program Entries (Internal Format on Disc), B-2
Program Marks (See End-of-File Marks)
PTA Command, 3-76

QUICK Command, 2-14, 3-76
QUICK Option, 1-10, 3-64, 3-76

RAE Command, 2-27, 3-77
RANDOM Command, 3-78
RECATALOG Command, 2-24, 3-79
Recataloging Programs in a USL, 2-24, 3-79
Recovering the WRK File, 3-78
REDEFINE Command, 3-80
REMOVE Command, 2-27, 3-81
Removing Programs, 2-27, 3-77, 3-81
RENAME Command, 2-27, 3-82
Renaming Programs, 2-27, 3-82
    Alias Names, 2-27, 3-100
    Renaming Attached Files, 3-80
REPLACE Command, 2-17, 3-83
Replacing Records in a Program, 2-17, 3-83
RESTORE Command, 2-41, 3-84
Restoring Saved USLs, 2-41, 3-84
REVISION Command, 3-85
Revision Strings, 3-85

SAVE Command, 2-41, 3-86
Saved Files (Internal Format on Disc), B-4
Saving USLs in Saved Format, 2-41, 3-86
SCALE Command, 3-87

SEARCH Command, 3-88
SEQUENCE Command, 2-35, 3-89
Sequence Numbers
    Checking Sequence Numbers for Ascending Order, 2-36, 3-13
    Generating Five-Digit Sequence Numbers, 2-35, 3-24
    Generating Four-Digit Sequence Numbers, 2-35, 3-89
    Inserting Sequence Numbers into Source Lines, 2-35
    Protecting Sequence Number Field, 2-36, 3-42
    Resetting Sequence Numbers, 2-37
    Stopping Sequence Numbers, 2-35, 3-61
SETTAB Command, 2-5, 3-90
SI File (Locating Programs on), 2-16, 3-72
SIZE Command, 3-91
SKIPTO Command, 2-13, 3-92
SLR Parameters, 2-10
SOURCE Command, 2-25, 3-94
Source Format for Program Listings, 2-25, 3-94
Source Line References, 2-10
    Character Strings, 2-11
    Current Line, 2-11
    Examples of, 2-12
    Line Numbers, 2-10
SPLIT Command, 3-96
Splitting a Source Line into Two Lines, 3-96
SQUEEZE Command, 2-40, 3-97
Subprograms, 2-28, 3-52

Tabs, 2-4
    Clearing Tabs, 2-5, 3-13
    Displaying a Scale with Tab Settings, 3-87
    Options for Processing Tabs, 3-64
    Redefining the Tab Character, 2-5, 3-17
    Setting Tabs, 2-5, 3-90
    Tabs used with the OVERLAY Command, 3-67
Terminator Character, 3-2
    For ADD and REPLACE Commands, 2-18, 3-5, 3-83
    Redefining the Terminator Character, 2-18, 3-17
TEST Command, 2-41, 3-98
Testing Saved USLs, 2-41, 3-98
TEXT Command, 2-25, 3-94, 3-98
Text Format for Program Listings, 2-25, 3-94, 3-98
TRANSFER Command, 2-31, 3-99
TWIN Command, 2-27, 3-100

UNCHECK Command, 2-37, 3-101
Uncompressed Records, 2-30, 3-58
UNDO Command, 3-101
UNSEQUENCE Command, 2-35, 3-102
USE Command, 3-102
USL File (Internal Format on Disc), B-3

VERIFY Command, 3-103
    Verify Errors, 2-37

Wildcard Characters
With the CAE Command, 3-9
With the LAE Command, 3-34
With the RAE Command, 2-27, 3-77
WRK File
Internal Format on Disc, B-4
Recovering Programs on the WRK File, 3-78
Use in Random Mode, 2-6

XLN Command, 2-26, 3-104
XRN Command, 3-106

YON Command, 3-28, 3-106

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 3624 FT. LAUDERDALE, FL 33309

POSTAGE WILL BE PAID BY ADDRESSEE

MODULAR COMPUTER SYSTEMS
1650 W. McNAB ROAD
P.O. BOX 6099
FT. LAUDERDALE, FLORIDA 33310

Attention: TECHNICAL PUBLICATIONS, M.S. #85

Fold

MODCOMP®

Please comment on the publication's completeness, accuracy, and readability. We also appreciate any general suggestions you may have to improve this publication.

If you found errors in this publication, please specify the page number or include a copy of the page with your remarks.

Your comments will be promptly investigated and appropriate action will be taken. If you require a written answer, please check the box and include your address below.

☐

Comments: _____

Manual Title _____

Manual Order Number _____ Issue Date _____

Name _____ Position _____

Company _____

Address _____

_____ Telephone ( ) _____

MCS 148D (1/83)

MODCOMP

Corporate Headquarters:
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310, Tel: (305) 974-1380, TWX: 310-372-7837

International Headquarters:
MODULAR COMPUTER SERVICES, Inc., The Business Centre, Molly Millars Lane, Wokingham, Berkshire, RG11 2JQ, UK, Tel: 0734-786808, TLX: 851849149

Latin American Sales Headquarters:
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310, Tel: (305) 975-6562, TLX: 3727652

Canadian Headquarters:
MODCOMP Canada, Ltd., 400 Matheson Blvd. East, Unit 24, Mississauga, Ontario, Canada L4Z 1N8, Tel: (416) 890-0666, TELEX: 06-961279

SALES & SERVICE LOCATIONS THROUGHOUT THE WORLD

"The technical contents of this document, while accurate as of the date of publication, are subject to change without notice."

Printed in the U.S.A.